# An Intelligent Internet Search Assistant based on the Random Neural Network

Will Serrano

Intelligent Systems and Networks Group, Electrical and Electronic Engineering

Imperial College London

[2] g.serrano11@imperial.ac.uk

**Abstract.** Web users can not be guaranteed that the results provided by Web search engines or recommender systems are either exhaustive or relevant to their search needs. Businesses have the commercial interest to rank higher on results or recommendations to attract more customers while Web search engines and recommender systems make their profit based on their advertisements and product purchase. This research analyses the result rank relevance provided by the different Web search engines, metasearch engines, academic databases and recommender systems. We propose an Intelligent Internet Search Assistant (ISA) that acts as an interface between the user and the different search engines. We also present a new relevance metric which combines both relevance and rank. We use this metric to validate and compare the performance of our proposed algorithm against other search engines and recommender systems. On average, our ISA outperforms other search engines

**Keywords:** Intelligent Internet Search Assistant; World Wide Web; Random Neural Network; Web search; Search Engines

## 1 Introduction

The need to search for specific information or products in the ever expanding Internet [28, 29] has led the development of Web search engines and recommender systems. Whereas their benefit is the provision of a direct connection between users and the information or products sought, any search outcome will be influenced by a commercial interest as well as by the users' own ambiguity in formulating their requests or queries. An example of this situation is travel services. The Internet has made accessible real time travel industry's information and services; customers can purchase flight tickets, hotels and holiday packs online. Distribution costs have been reduced due a shorter value chain; however businesses not shown on the top positions within the search results may lose potential customers. A similar scenario also occurs within academic search; the Internet has allowed the democratization of academic publications. Authors can upload their work onto their personal Webpages bypassing the traditional model of the journal peer review. There is the biased interest from authors to get their publications in top search positions in order to reach a bigger audience so

they will be cited more. In both examples ranking algorithms are essential as they decide the relevance; they make information visible or hidden to customers or users. Under this model, Web search engines or recommender systems can be tempted to artificially rank results from some specific businesses for a fee whereas also authors or business can be tempted to manipulate ranking algorithms by "optimizing" the presentation of their work or products. The main consequence is that irrelevant results may be shown on top positions and relevant ones "hidden" at the very bottom of the search list.

We describe the application of neural networks in Web search and recommender systems in Section 2. In order to address the presented search issues; this paper proposes in Section 3 an Intelligent Internet Search Assistant (ISA) that acts as an interface between an individual user's query and the different search engines. Our ISA acquires a query from the user and retrieves results from one or various search engines assigning one neuron per each Web result dimension. The result relevance is calculated by applying our innovative cost function based on the division of a query into a multidimensional vector weighting its dimension terms with different relevance parameters. Our ISA adapts and learns the perceived user's interest and reorders the retrieved snippets based in our dimension relevant centre point. Our ISA learns result relevance on an iterative process where the user evaluates directly the listed results. We evaluate and compare its performance against other search engines with a new proposed quality definition, which combines both relevance and rank. We have also included two learning algorithms; Gradient Descent learns the centre of relevant dimensions and Reinforcement Learning updates the network weights based on rewarding relevant dimensions and punishing irrelevant ones. We have validated our ISA against other Web search engines and metasearch engines, online databases and recommender systems on Section 4. We have also analysed the Gradient Descent and Reinforcement Learning algorithms based on result relevance and learning speed; our conclusions are presented on Section 5.

## 2 Related work

The ability of neural networks to learn iteratively from different inputs to acquire the desired outputs as a mechanism of adaptation to users' interest in order to provide relevant answers have already been applied in the World Wide Web and recommender systems. Scarselli, F. et al [1] and Chau, M. et al [2] use a neural network by assigning a neuron to each Web page; therefore they create a graph where the neural links are the equivalent of the hyperlinks; in our proposal we have assigned a neuron per result dimension optimizing its ranking based on a Random Neural Network with a defined cost function. Bermejo, S. et al [3] use a similar approach to our proposal, the allocation of one neuron per Web search result, however the main difference is that the network is trained to cluster results by meaning; we reorganize results without any previous training by using the Random Neural Network algorithm iteratively with a defined cost function. Burgues, C. et al [4] define RankNet which uses neural net-

works to evaluate Web sites by training the neural network based on query-document pairs. In our solution it is the user who recursively trains the network while selecting relevant results. Shu, B. et al [5] retrieve results from different Web search engines and train the network following the belief that a result in a top position would be relevant, the main difference with our research is that we use a cost function to reorganize results and learn the user's perception of relevance. Boyan, J. et al [6] use Reinforcement Learning to rank Web pages using their HTML properties and hyperlink connections between them; it differs from our approach that consists of ranking the results provided by Web search engines. Wang, X. et al [7] use a back propagation neural network with its input nodes corresponding to an specific quantified user profile and one output node which it is the a probability the user would consider the Web page relevant whereas we assign a neuron per result dimension instead of user profile term.

S. Patil et al [8] propose a recommender system using collaborative filtering mechanism with k-separability approach for Web based marketing. They build a model for each user on several steps: they cluster a group of individuals into different categories according to their similarity using Adaptive Resonance Theory (ART) and then they calculate the Singular Value Decomposition matrix. They use a feed forward neural network where the input is the user ratings' matrix and the target user and the output is the user model. M. Lee et al [9] propose a new recommender system which combines collaborative filtering with a Self-Organizing Map neural network. They segment all users by demographic characteristics where users in each segment are clustered according to the preference of items using the neural network. The collaborative filtering algorithm is applied on the cluster where the user belongs. The Self-Organizing Map neural network is an unsupervised learning model which learns the preference of items in each segment. Its input is the user segments and the output is the cluster type. It provides to the collaborative filter algorithm the cluster the user belongs to. C. Vassiliou et al [10] propose a framework that combines neural networks and collaborative filtering. Their approach uses a neural network to recognize implicit patterns between user profiles and items of interest which are then further enhanced by collaborative filtering to personalized suggestions. The neural network algorithm is trained on each user ratings vector. The output of the neural network is a pseudo user ratings vector that fills the unrated items; this avoids the common sparsity issue on recommender systems. The neural network is a multiplayer feed forward model. K. Kongsakun et al [11] develop an intelligent recommender system framework based on an investigation of the possible correlations between the students' historic records and final results. The have used a multi layered feed forward neural network to find structures and relationships within the data with a supervised learning process. C. Chang et al [12] train the artificial neural networks to group users into different types. They use an Adaptive Resonance Theory (ART) neural network model in an unsupervised learning model where the input layer is a vector made of user's features and the output layer is the different cluster. The ART neural network is formed of 255 input and 5 output neurons. P. Chou et al [13] integrate a back propagation neural network with supervised learning and feed forward architecture in an "interior desire system". The rationale of the proposed approach is that if users have similar navigation patterns,

then they may have similar interest for some products. The neural network classifies users with similar navigation patterns into groups with similar intention behavioural patterns. D. Billsus et al [14] propose a representation for collaborative filtering tasks that allows the application of any machine learning algorithm, including a feed forward neural network with k input neurons, 2 hidden neurons and 1 output neuron. They convert the training data, a sparse matrix of user ratings to boolean feature vectors and then they calculate the Singular Value Decomposition (SVD). They train the neural network with the k singular vector. The output neuron represents the predicted user rating. M. Krstic et al [15] apply a single hidden layer feed forward neural network as a classifier tool which estimates whether a certain TV programme is relevant to the user based on the TV programme description, contextual data and the feedback provided by the user. The number of neurons is the input layer is determined by the number of dimensions of the vector space, three for the genre coordinates and two for contextual data. The number of output neuros is two, one for like and the other for dislike. C. Biancalana et al [16] propose a neural network to include contextual information on film recommendations. The aim of the neural network is to identify which member of a household gave a specific rating to a film at a specific time. The input layer is formed of 68 neurons and the output layer consists of 3 neurons which represent the different classifiers. M. Devi et al [17] use a probabilistic neural network to calculate the rating between users based on the rating matrix. They smooth the sparse rating matrix by predicting the rating values of the unrated items. They model users using a Self-Organizing Map and unsupervised techniques. The probabilistic neural network is used to identify the rating cluster.

## 3 The Intelligent Internet Search Assistant Model

The search assistant we design is based on the Random Neural Network (RNN) [18, 19,20]. This is a spiking recurrent stochastic model for neural networks. Its main analytical properties are the "product form" and the existence of the unique network steady state solution. It represents more closely how signals are transmitted in many biological neural networks where they actual travel as spikes or impulses, rather than as analogue signal levels, and has been used in different applications including network routing with cognitive packet networks, using reinforcement learning, which requires the search for paths that meet certain pre-specified quality of service requirements [21], search for exit routes for evacuees in emergency situations [22,23] and network routing [27], pattern based search for specific objects [24], video compression [25], image texture learning and generation [26].

In the case of our own application of the RNN, the search for information or for some meaning needs requires us to specify some elements: an M-dimensional universe of X entities or ideas to be searched, a high level query that specifies the N-properties or concepts requested by a user and a method that searches and selects Y entities from the universe showing the first Z results to user according to an algorithm or rule. Each entity or concept in the universe is distinct from the others in some recognizable way;

for instance two entities may be different just in the date or time-stamp that characterizes the time when they were last stored or in the ownership or origin of the entities. On the other hand, we consider concepts to be distinct if they contain any different meaning, even though if they are identical with respect to a user's query.

We consider that the universe which we are searching within as a relation U that consists of a set of X M-tuples, $U = \{v_1 , v_2 \dots v_X\}$, where $v_i = (l_{i1} , l_{i2} \dots l_{iM})$ and li are the M different attributes for i=1,2..X. The relation U is a very large relation consisting on M >> N attributes. The important concept in the development of this paper is a query can be defined as $R_t(n(t)) = (R_t(1), R_t(2), \dots, R_t(n(t)))$ where n(t) is a variable N-dimension attribute vector with $1<N<M$ and t is the search iteration being $t>0$; n(t) is variable so that attributes can be added or removed based on their relevance as the search progresses, i.e. as t increases. Each $R_t(n(t))$ takes its values from the attributes within the domain D(n(t)), where D is the corresponding domain that forms the universe U. Thus D(n(t)) is a set of properties or meanings based in words or integers, but also words in another language, or a set of icons, images or sounds.

The answer A to the query $R_t(n(t))$ is a set of Y M-tuples $A = \{v_1 , v_2 \dots v_Y\}$ where $v_o = (l_{o1} , l_{o2} \dots l_{oM})$ and lo are the M different attributes for o=1,2..Y. Our Intelligent Internet Search Assistant only shows to the user the first set of Z tuples that have the highest neuron potentials among the set of Y tuples. The neuron potential that represents the relevance of each M-tuple $v_o$ is calculated at each t iteration. The user or the high level query itself is limited mainly by two main factors: the user's lack of information about all the attributes that form the universe U of entities and ideas, or the user's lack of precise knowledge about what he is looking for.

### 3.1 Result Cost Function

We consider the universe U is formed of the entire results that can be searched. We assign each result provided by a search engine to an M-tuple $v_o$ of the answer set A. We calculate the result relevance based on a cost function described within this section. The query $R_t(n(t))$ is a variable N-dimension vector that specifies the attributes the user consider relevant. The number of dimensions of the attribute vector n(t) varies as the iteration t increases. Our Intelligent Internet Search Assistant associates an M-tuple $v_o$ to each result provided by the Search Engine creating an answer set A of Y M-tuples. Search Engines select their results from the universe U. We apply our cost function to each result or M-tuple $v_o$ from the answer set A of Y M-tuples. We consider each $v_o$ as a M-dimensional vector. The cost function is firstly calculated based on the relevant N attributes the user introduced on the High Level Query, $R_1(n(1))$ within the domain D(n(1)) however, as the search progresses, $R_t(n(t))$, attributes may be added or removed based on the perceived relevance within the domain D'(n(t)). We calculate the overall Result Score, RS, by measuring the relationship between the values of its different attributes:

$$RS = RV * HW \tag{1}$$

where RV is the Result Value which measures the result relevance and HW the Homogeneity Weight. The Homogeneity Weight (HW) rewards results that have relevance or scores dispersed along their attributes. This parameter is also based on the idea that the first dimensions or attributes of the user query $R_t(n(t))$ are more important than the last ones:

$$HW = \frac{\sum_{n=1}^{N} HF[n]}{N} \tag{2}$$

where HF[n], homogeneity factor, is a N-dimension vector associated to the result and n is the attribute index from the query $R_t(n(t))$:

$$HF[n] = \frac{N - n}{N} \quad \text{if } SD[n] > 0 \quad \Big| \quad HF[n] = 0 \quad \text{if } SD[n] = 0 \tag{3}$$

We define Score Dimension SD[n] as a N-dimension vector that represents the attribute values of each result or M-tuple $v_o$ in relation with the query $R_t(n(t))$. The Result Value (RV) is the sum of each dimension individual score:

$$RV = \sum_{n=1}^{N} SD[n] \tag{4}$$

where n is the attribute index from the query $R_t(n(t))$. Each dimension of the Score Dimension vector SD[n] is calculated independently for each n-attribute value that forms the query $R_t(n(t))$:

$$SD[n] = S * PPW * RPW * DPW \tag{5}$$

We consider only three different types of domains of interest: words, numbers (as for dates and times) and prices. S is the score calculated depending if the domain of the attribute is a word (WS), number (NS) or price (PS). If the domain D(n) is a word, our ISA calculates the score Word Score (WS) following the formula:

$$S = \frac{WR}{NW} \tag{6}$$

where the value of WR is 1 if the word of the n-attribute of the query $R_t(n(t))$ is contained in the search result or 0 otherwise. NW is the number of words in the search result. If the domain D(n) is a number, our ISA selects the best Number Score (NS) from the numbers they are contained within the search result that maximizes the cost function:

$$S = \frac{\left(1 - \left(\frac{|DV - RV|}{|DV| + |RV|}\right)\right)}{NN} \tag{7}$$

where DV is the value of the n-attribute of the query $R_t(n(t))$, RV is the value of a number in the result and NN is the total number of numbers in the result. If the domain D(n) is a price, our ISA chooses the best Price Score (PS) from the prices in the result that maximizes the cost function:

$$S = \frac{\left(\frac{DV}{RV}\right)}{NP} \tag{8}$$

where DV is value of the n-attribute of the query $R_t(n(t))$, RV is the value of a price in the result and NP is the total number of prices in the result. We penalize if the search result provides unnecessary information by dividing the score by the total amount of elements in the Web result. The dimension Score Dimension vector, SD[n] is weighted according to different relevance factors:

$$SD[n] = S * PPW * RPW * DPW \tag{9}$$

The Position Parameter Weight (PPW) is based on the idea that an attribute value shown within the first positions of the search result is more relevant than if it is shown at the final:

$$PPW = \frac{NC - DVP}{NC} \tag{10}$$

where NC is the number of characters in the result and DVP is the position within the result where the value of the dimension is shown. The Relevance Parameter Weight (RPW) incorporates the user's perception of relevance by rewarding the first attributes of the query $R_t(n(t))$ as highly desirable and penalising the last ones:

$$RPW = 1 - \frac{PD}{N} \tag{11}$$

where PD is the position of the n-attribute of the query $R_t(n(t))$ and N is the total number of dimensions of the query vector $R_t(n(t))$. The Dimension Parameter Weight (DPW) incorporates the observation of user relevance with the value of domains D(n(t)) by providing a better score on the domain values the user has more filled on the query:

$$DPW = \frac{NDT}{N} \tag{12}$$

where NDT is the number of dimensions with the same domain (word, number or price) on the query $R_t(n(t))$ and N is the total number of dimensions of the query vector $R_t(n(t))$. We assign this final Result Score value (RS) to each M-tuple $v_o$ of the answer set A. This value is used by our ISA to reorder the answer set A of Y M-tuples, showing to the user the first set of Z results which have the higher potential.

### 3.2 User Iteration

The user, based on the answer set A can now act as an intelligent critic and select a subset of P relevant results, $C_P$, of A. $C_P$ is a set that consists of P M-tuples $C_P = \{v_1 , v_2 \dots v_P\}$. We consider $v_P$ as a vector of M dimensions; $v_p = (l_{p1} , l_{p2} \dots l_{pM})$ where $l_p$ are the M different attributes for p=1,2..P. Similarly, the user can also select a subset of Q irrelevant results, $C_Q$ of A, $C_Q = \{v_1 , v_2 \dots v_Q\}$. We consider $v_q$ as a vector of M dimensions; $v_q = (l_{q1} , l_{q2} \dots l_{qM})$ where lq are the M different attributes for q=1,2..Q. Based on the user iteration, our Intelligent Internet Search Assistant provides to the user with a different answer set A of Z M-tuples reordered to MD, the minimum distance to the Relevant Centre for the results selected, following the formula:

$$RCP[n] = \frac{\sum_{p=1}^{P} SD_p[n]}{P} = \frac{\sum_{p=1}^{P} l_{pn}}{P} \tag{13}$$

where P is the number of relevant results selected, n the attribute index from the query $R_t(n(t))$ and $SD_p[n]$ the associated Score Dimension vector to the result or M-tuple $v_P$ formed of $l_{pn}$ attributes. An equivalent equation applies to the calculation of the Irrelevant Centre Point. Our Intelligent Internet Search Assistant reorders the retrieved Y set of M-tuples showing only to the user the first Z set of M-tuples based on the lowest distance (MD) between the difference of their distances to both Relevant Centre Point (RD) and the Irrelevant Centre Point (ID) respectively:

$$MD = RD - ID \tag{14}$$

where MD is the result distance, RD is the Relevant Distance and ID is the Irrelevant Distance. The Relevant Distance (RD) of each result or M-tuple $v_q$ is formulated as:

$$RD = \sqrt{\sum_{n=1}^{N} (SD[n] - RCP[n])^2} \tag{15}$$

where SD[n] is the Score Dimension vector of the result or M-tuple $v_q$ and RCP[n] is the coordinate of the Relevant Centre Point. Equivalent equation applies to the calculation of the Irrelevant Distance. Therefore we are presenting an iterative search progress that learns and adapts to the perceived user relevance.

### 3.3 Dimension Learning

The answer set A to the query $R_1(n(1))$ is based on the N dimension query introduced by the user however results are formed of M dimensions therefore the subset of results the user has considered as relevant may have other relevant concepts hidden the user did not considered on the original query. We consider the domain D(m) or the M attributes from which our universe U is formed as the different independent words that form the set of Y results retrieved from the search engines. Our cost function is ex-

panded from the N attributes defined in the query $R_1(n(1))$ to the M attributes that form the searched results. Our Score Dimension vector, SD[m], is now based on M-dimensions. An analogue attribute expansion is applied to the Relevance Centre Calculation, RCP[m]. The query $R_1(n(1))$ is based on the N-Dimension vector introduced by the user however the answer set A consist of Y M-tuples. The user, based on the presented set A, selects a subset of P relevant results, $C_P$ and a subset of Q irrelevant results, $C_Q$.

Lets consider $C_P$ as a set that consists of P M-tuples $C_P = \{v_1 , v_2 \dots v_P\}$ where $v_P$ is a vector of M dimensions; $v_P = (l_{p1} , l_{p2} \dots l_{pM})$ and $l_p$ are the M different attributes for p=1,2..P. The M-dimension vector Dimension Average, DA[m], is the average value of the m-attributes for the selected relevant P results:

$$DA[m] = \frac{\sum_{p=1}^{P} SD_p[m]}{P} = \frac{\sum_{p=1}^{P} l_{pm}}{P} \tag{16}$$

where P is the number of relevant results selected, m the attribute index of the relation U and $SD_p[m]$ the associated Score Dimension vector to the result or M-tuple $v_P$ formed of $l_{pm}$ attributes. We define ADV as the Average Dimension Value of the M-dimension vector DA[m]:

$$ADV = \frac{\sum_{m=1}^{M} DA[m]}{M} \tag{17}$$

where M is the total number of attributes that form the relation U. The correlation vector σ[m] is the difference between the dimension values of each result with the average vector:

$$\sigma[m] = \frac{\sum_{p=1}^{P} \left(SD_p[m] - DA[m]\right)}{P} = \frac{\sum_{p=1}^{P} \left(l_{Pm} - DA[m]\right)}{P} \tag{18}$$

where P is the number of relevant results selected, m the attribute index of the relation U and $SD_p[m]$ the associated Score Dimension vector to the result or M-tuple $v_P$ formed of $l_{pm}$ attributes. We define C as the average correlation value of the M-dimensions of the vector σ[m]:

$$C = \frac{\sum_{m=1}^{M} \sigma[m]}{M} \tag{19}$$

where M is the total number of attributes that form the relation U. We consider an m-attribute relevant if its associated Dimension Average value DA[m] is larger than the average dimension ADV and its correlation value σ[m] is lesser than the average cor-

relation C. We have therefore changed the relevant attributes of the searched entities or ideas by correlating the error value of its concepts or properties represented as attributes or dimensions. On the next iteration, the query $R_2(n(2))$ is formed by the attributes our ISA has considered relevant. The answer to the query $R_2(n(2))$ is a different set A of Y M-tuples. This process iterates until there are not new relevant results to be shown to the user.

### 3.4 Gradient Descent Learning

Gradient Descent learning is based on the adaptation to the perceived user interests or understanding of meaning by correlating the attribute values of each result to extract similar meanings and cancel superfluous ones. The ISA Gradient Descent learning algorithm is based on a recurrent model. The inputs $i = \{i_1,\ldots,i_P\}$ are the M-tuples $v_P$ corresponding to the selected relevant result subset $C_P$ and the desired outputs $y = \{y_1,\ldots,y_P\}$ are the same values as the input. Our ISA then obtains the learned random neural network weights, calculates the relevant dimensions and finally reorders the results according to the minimum distance to the new Relevant Centre Point focused on the relevant dimensions.

### 3.5 Reinforcement Learning

The external interaction with the environment is provided when the user selects the relevant result set $C_P$. Reinforcement Learning adapts to the perceived user relevance by incrementing the value of relevant dimensions and reducing it for the irrelevant ones. Reinforcement Learning modifies the values of the m attributes of the results, accentuating hidden relevant meanings and lowering irrelevant properties. We associate the Random Neural Network weights to the answer set A; W = A. Our ISA updates the network weights W by rewarding the result relevant attributes by:

$$w(p,m) = l_{pm}^{s-1} + l_{pm}^{s-1} * \left( \frac{l_{pm}^{s-1}}{\sum_{m=1}^{M} l_{pm}^{s-1}} \right) \tag{20}$$

where p is the result or M-tuple $v_P$ formed of $l_{pm}$ attributes, m the result attribute index, M the total number of attributes and s the iteration number. ISA also updates the network weights by punishing the result irrelevant attributes by:

$$w(p,m) = l_{pm}^{s-1} - l_{pm}^{s-1} * \left( \frac{l_{pm}^{s-1}}{\sum_{m=1}^{M} l_{pm}^{s-1}} \right) \tag{21}$$

where p is the result or M-tuple $v_P$ formed of $l_{pm}$ attributes, m the result attribute index, M the total number of attributes and s the iteration number. Our ISA then recalculates the potential of each of the result based on the updated network weights and reorders them, showing to the user the results which have a higher potential or score.

# 4 Validation

We can affirm the superior search engine will have the higher density of better scoring results on top positions based on the result master list. In order to measure numerically Web search quality or to establish a benchmark from we can compare Web search performance; we propose the following algorithm, where results showed at top positions are rewarded and results showed at lower positions are penalized. Lets define quality, Q, as:

$$Q = \sum_{result=1}^{Y} RML * RSE \tag{22}$$

where RML is the rank of the result in the master list, RSE is the rank of the same result in a particular search engine and Y is the number of results shown to the user, if the result order is larger than Y, we discard the result in our calculation as it is considered irrelevant. We define normalized quality, $\overline{Q}$ , as the division of the quality, Q, by the optimum figure which it is when the results provided are ranked in the same order as in the master list; this value corresponds to the sum of the squares of the first Y integers:

$$\overline{Q} = \frac{Q}{\frac{Y(Y+1)(2Y+1)}{6}} \tag{23}$$

where Y the total number of results shown to the user.

## 4.1 Web Search Validation

Intelligent Internet Search Assistant we have proposed emulates how Web search engines work by using a very similar interface to introduce and display information. We validate our proposed ISA with current Metasearch engines, we retrieve the results from the Web search engines they use to generate the result master list and then compare the results provided by the Metasearch engines against this result master list. This proposed method has the inconvenience that we are not considering any result obtained from Internet Web directories neither Online databases from where Metasearch engines may have retrieved some results displayed. We have selected both Ixquick and Metacrawler as the Metasearch engines we can compare our ISA. After analysing the main characteristics of both Metasearch engines we consider Metacrawler uses (Google Yahoo and Yandex) and Ixquick uses (Google Yahoo and Bing) as their main source of search results. We have run our ISA to acquire 10 different high level queries based on the travel industry from a user. The ISA then retrieves the first 30 results from each of the main Web search engine driver programmed (Google, Yahoo, Bing, and Yandex), we have therefore scored 30 points to the Web site result that is displayed in the top position, 1 point to the Web site result that is shown in the last position and 0 points to each of the result that belongs to the same Web site and it is shown more than once. After we have scored the 120 results provided by the 4 different Web search engines, we combine them by adding the scores of the results which have the same Web site and rank them to generate the result master list. We have done this

evaluation exercise for each high level query. We then retrieve the first 30 results from Metacrawler and Ixquick and benchmark them against the result master list using the Quality formula proposed. We present the average Quality values for the 10 different queries on the below table.

**Table 1.** Web Search Validation

| 10 Queries | | | | | | |
|---|---|---|---|---|---|---|
| **Google** | **Yahoo** | **Bing** | **Yandex** | **MetaC** | **Ixquick** | **ISA** |
| 0.67 | 0.66 | 0.66 | 0.68 | 0.59 | 0.42 | 0.65 |

### 4.2 Database Validation

Our Intelligent Internet Search Assistant can select between the main Online Academic (Google Scholar, IEEE Xplore, CiteseerX or Microsoft Academic) and the type of learning to be implemented. Our ISA then collects the first 50 results from the search engine selected, reorders them according to its cost function and finally shows to the user the first 20 results. Our ISA reorders results while learning on the two step iterative process showing only the best 20 results to the user. We have searched for 6 different queries. We have used the four different Online Academic Databases for each query, 24 searches in total. We have selected Gradient Descent and Reinforcement Learning for 3 queries (12 searches) each. The table shows the average Quality value of the Database search engine and ISA. The first I represents the improvement from ISA against the Online Academic Databases; the second I is between ISA iterations 2 and 1 and finally the third I is between the ISA iterations 3 and 2.

**Table 2.** Database Validation

| Gradient Descent Learning: 12 Queries | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Web** | **ISA** | **I** | **Web** | **ISA** | **I** | **Web** | **ISA** | **I** |
| 0.44 | 0.56 | 29% | 0.48 | 0.64 | 13% | 0.50 | 0.66 | 3.6% |
| **Reinforcement Learning: 12 Queries** | | | | | | | | |
| **Web** | **ISA** | **I** | **Web** | **ISA** | **I** | **Web** | **ISA** | **I** |
| 0.41 | 0.51 | 25% | 0.44 | 0.61 | 19% | 0.46 | 0.64 | 5.2% |

### 4.3 Recommender System Validation

We have implemented our Intelligent Search Assistant to reorder the results from three different independent recommender systems: GroupLens film database, Trip Advisor and Amazon. Our ISA reorders the films or products based on the updated result relevance calculated by combining only the value of the relevant selected dimensions. The higher the value the more relevant the film or product should be. ISA shows to the user the first 20 results including its ranking. The user then selects the films or products with higher ranking; this ranking has been previously calculated by adding user reviews to the same products and calculating the average value. We have included Gradient Descent and Reinforcement Learning for different queries in our validation. The

table below show the average Quality value, the first I represents the improvement from ISA against the recommender system; the second I is between ISA iterations 2 and 1 and finally the third I is between the ISA iterations 3 and 2.

**Table 3.** Recommder System Validation

| **GroupLens film dataset - Gradient Descent Learning -5 Queries** | | | |
|---|---|---|---|
| **First (Q)** | **Iteration 1 (I)** | **Iteration 2 (I)** | **Iteration 3 (I)** |
| 0.71 | 17.45% | 0.145% | 1.79% |
| **GroupLens film dataset - Reinforcement Learning -5 Queries** | | | |
| 0.76 | 5.26% | 9.14% | 6.05% |
| **Trip advisor car dataset - Gradient Descent Learning -5 Queries** | | | |
| 0.94 | 0.0328% | 0.017% | 0.0007% |
| **Trip advisor car dataset - Reinforcement Learning -5 Queries** | | | |
| 0.94 | 0.798% | 0.004% | 0.0165% |
| **Trip advisor hotel dataset-Gradient Descent Learning -5 Queries** | | | |
| 0.94 | 0.54% | -0.0607% | -0.0395% |
| **Trip advisor hotel dataset - Reinforcement Learning -5 Queries** | | | |
| 0.94 | 0.728% | 4.658% | 0.139% |
| **Amazon dataset-Gradient Descent Learning -5 Queries** | | | |
| 0.15 | 33.89% | 8.39% | -6.97% |
| **Amazon dataset - Reinforcement Learning -5 Queries** | | | |
| **First (Q)** | **Iteration 1 (I)** | **Iteration 2 (I)** | **Iteration 3(I)** |
| 0.15 | 30.36% | 13.05% | 0.503% |

# 5 Conclusions

We have proposed a novel approach to Web search and recommendation systems where the user iteratively trains the neural network while looking for relevant results. We have also defined a different process; the application of the Random Neural Network as a biological inspired algorithm to measure both user relevance and result ranking based on a predetermined cost function. Our Intelligent Search Assistant performs generally slightly better than Google and other Web search engines however, this evaluation may be biased because users tend to concentrate on the first results provided which were the ones we showed in our algorithm. Our ISA adapts and learns from user previous relevance measurements increasing significantly its quality and improvement within the first iteration. Reinforcement Learning algorithm performs better than Gradient Descent. Although Gradient Descent provides a better quality on the first iteration; Reinforcement Learning outperforms on the second one due its higher learning rate. Both of them have a residual learning on their third iteration. Gradient Descent would have been the preferred learning algorithm if only one iteration is required; however Reinforcement Learning would have been a better option in

the case of two iterations. It is not recommended three iterations because learning is only residual.

## Acknowledgment

This research has used Groupfilms dataset from the Department of Computer Science and Engineering at the University of Minnesota; Trip Advisor dataset from the University of California-Irvine, Machine Learning repository, Centre for Machine Learning and Intelligent Systems and Amazon dataset from Julian McAuley Computer Science Department at University of California, San Diego.

## References

1. Scarselli, F., Liang, S., Hagenbuchner, M., Chung, A.: Adaptive page ranking with neural networks. *Proceeding WWW '05 Special interest tracks and posters of the 14th international conference on World Wide Web*, 936- 937 (2005)
2. Chau, M., Chen, H.: Incorporating Web analysis into neural networks: an example in Hopfield net searching. *IEEE transactions on systems and cybernetics – Part C: applications and reviews*, Vol 37, No 3, 352-358 (2007)
3. Bermejo, S., Dalmau, J.: Web metasearch using unsupervised neural networks. *IWANN '03 Proceedings of the 7th International work-conference on artificial and natural neural networks: Part II: artificial neural nets problem solving methods*, 711-718 (2003)
4. Burgues, C., Shaked, T., Renshaw, E., Lazier, L., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. *ICML '05 Proceedings of the 22nd international conference on machine learning*, 89-96 (2005)
5. Shu, B., Kak, S.: A neural network-based intelligent metasearch engine. *Information sciences, informatics and computer science*, Vol 120, 1-11 (2009)
6. Boyan, J., Freitag, D., Joachims, T.: A machine learning architecture for optimizing Web search engines. *Proceedings of the AAAI workshop on Internet-based information systems* (1996)
7. Wang, X., Zhang, L.: Search engine optimization based on algorithm of BP neural networks. *Proceedings of the seventh international conference on computational intelligence and security*, 390-394 (2011)
8. Patil, S., Mane, Y., Dabre, K., Dewan, P. and Kalbande, D.: An efficient recommender system using collaborative filtering methods with k-separability approach. *International journal of engineering research and applications*, 30-35 (2012)
9. Lee, M., Choi, P., Woo, Y.: A hybrid recommender system combining collaborative filtering with neural network. *Adaptive hypermedia and adaptive Web-based systems,* Vol 2347, 531-534 (2002)
10. Vassiliou, C., Stamoulis, D., Martakos, D., Athanassopoulos, S.: A recommender system framework combining neural networks & collaborative filtering. *International conference on instrumentation, measurement, circuits and systems*, 285-290 (2006)
11. Kongsakun, K., Kajornrit J., Fung, C.: Neural network modelling for an intelligent recommendation system supporting SRM for universities in Thailand. *International conference on computing and information technology*, Vol 2, 34-44 (2013)

12. Chang, C., Chen, P., Chiu F., Chen, Y.: Application of neural networks and Kanos's method to content recommendation in Web personalization. *Expert systems with applications,* Vol 36, 5310-5316 (2009)
13. Chou, P., Li, P., Chen, K., Wu, M.: Integrating Web mining and neural network for personalized e-commerce automatic service. *Expert Systems with applications*, Vol 37, 2898-2910 (2010)
14. Billsus, D., Pazzani, M.: Learning collaborative information filters. *International conference of machine learning,* 46-54 (1998)
15. Krstic, M., Bjelica, M.: Context aware personalized program guide based on neural network. *IEEE transactions on consumer electronics*, Vol 58, 1301-1306 (2012)
16. Biancana, C., Gaspareti, F., Micarelli, A., Miola A., Sansonetti, G.: Context-aware movie recommendation based on signal processing and machine learning. *The challenge on context aware movie recommendation*, 5-10 (2011)
17. Devi, M., Samy, R., Kumar, S., Venkatesh, P.: Probabilistic neural network approach to alleviate sparsity and cold start problems in collaborative recommender systems. *Computational intelligence and computing research,* 1-4 (2010)
18. Gelenbe, E.: Random neural network with negative and positive signals and product form solution" *Neural Computation* 1, 502-510 (1989)
19. Gelenbe, E.: Learning in the recurrent Random Neural Network. *Neural Computation.* 5, 154-164 (1993)
20. Gelenbe, E., Timotheou, S.: Random neural networks with synchronized interactions. *Neural Computation*, 20(9): 2308 – 2324 (2008)
21. Gelenbe, E., Lent, R., Xu, Z.: Towards networks with cognitive packets. Performance and QoS of next generation networking. pp 3-17, Springer (London), (2011)
22. Gelenbe, E., Wu, F.J.: Large scale simulation for human evacuation and rescue. Computers & Mathematics with Applications 64 (12), 3869-3880 (2012)
23. Filippoupolitis, A., Hey, L., Loukas, G., Gelenbe, E., Timotheou, S.: Emergency response simulation using wireless sensor networks. Proceedings of the 1st international conference on Ambient media and systems, 21, (2008)
24. Gelenbe, E., Koçak, T.: Area-based results for mine detection. Geoscience and Remote Sensing, IEEE Transactions on 38 (1), 12-24 (2000)
25. Cramer, C., Gelenbe, E., Bakircloglu, H.: Low bit-rate video compression with neural networks and temporal subsampling. Proceedings of the IEEE 84 (10), 1529-1543 (1996)
26. Atalay, V., Gelenbe, E., Yalabik, N.: The random neural network model for texture generation. International Journal of Pattern Recognition and Artificial Intelligence, 6 (1):131-141 (1992)
27. Gelenbe, E.: Steps towards self-aware networks. Communications of the ACM, 52 (7): 66-75 (2009)
28. Gelenbe, E.: Search in unknown random environments. PHYSICAL REVIEW E 82 (6), 061112 (2007)
29. Gelenbe, E. and Abdelrahman, O. H.: Search in the universe of big networks and data. IEEE Network 28(4): 20-25 (2014)
30. Abdelrahman, O. H. and Gelenbe, E.: Time and energy in team-based search. PHYSICAL REVIEW E 87, 032125 (2013)