European Review, Vol. 25, No. 2, 260–272 © 2017 Academia Europæa
doi:10.1017/S1062798716000594

Smart Internet Search with Random Neural Networks

WILL SERRANO

Intelligent Systems and Networks Group, Electrical and Electronic Engineering, Imperial College London, UK. E-mail: g.serrano11@imperial.ac.uk

Web services that are free of charge to users typically offer access to online information based on some form of economic interest of the web service itself. Advertisers who put the information on the web will make a payment to the search services based on the clicks that their advertisements receive. Thus, end users cannot know that the results they obtain from web search engines are exhaustive, or that they actually respond to their needs. To fill the gap between user needs and the information presented to them on the web, Intelligent Search Assistants have been proposed to act at the interface between users and search engines to present data to users in a manner that reflects their actual needs or their observed or stated preferences. This paper presents an Intelligent Internet Search Assistant based on the Random Neural Network that tracks the user's preferences and makes a selection on the output of one or more search engines using the preferences that it has learned. We also introduce a 'relevance metric' to compare the performance of our Intelligent Internet Search Assistant against a few search engines, showing that it provides better performance.

1. Introduction

The need to search for specific information in the ever-expanding internet[1–3] has led to the development of web search engines. 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 in real time the 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 searching; the internet has allowed the democratization of academic publications. Authors can

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

upload their work onto their personal webpages bypassing the traditional journal peer review model. There is biased interest from authors to get their publications into the 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 recommending systems may rank results from some specific businesses for a fee based on a form of auction,[4] whereas 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.

In order to address the presented search issues; this paper proposes 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 on our dimension relevant centre point. Our ISA learns result relevance though an iterative process where the user directly evaluates 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 using travel services and open user queries. We have also analysed the Gradient Descent and Reinforcement Learning algorithms based on result relevance and learning speed.

We describe the application of neural networks in web searching in Section 2. We define our Intelligent Internet Search Assistant mathematical model in Section 3 and we have validated it against other web search engines in Section 4. Finally, we present our conclusions in 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 has already been applied in the World Wide Web and recommender systems.

F. Scarselli *et al.*[5] and M. Chau *et al.*[6] use a neural network by assigning a neuron to each web page; they create a graph where the neural links are the equivalent of the hyperlinks. S. Bermejo *et al.*[7] 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. C. Burgues *et al.*[8] define RankNet, which uses neural networks to evaluate websites by training the neural network based on

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

query-document pairs. B. Shu *et al.*[9] retrieve results from different web search engines and train the network following the assumption that a result in a top position would be relevant. J. Boyan *et al.*[10] use reinforcement learning to rank web pages using their HTML properties and hyperlink connections between them. X. Wang *et al.*[11] 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 probability that the user would consider the web page relevant.

3. The Intelligent Internet Search Assistant Model

The search assistant we design is based on the Random Neural Network (RNN).[12–14] This is a biologically inspired 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. The RNN represents more closely how signals are transmitted in many biological neural networks where they actually travel as spikes or impulses, rather than as analogue signal levels. It 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,[15,16] search for exit routes for evacuees in emergency situations,[17,18] pattern-based search for specific objects,[19] video compression,[20] and image texture learning and generation.[21]

3.1. Search Model

In the case of our own application of the RNN, the search for information or for some meaning 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 the 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 they are identical with respect to a user's query.

We consider the universe we are searching within as a relation U that consists of a set of X M-tuples, $U = \{v_1, v_2, \ldots, v_X\}$, where $v_i = (l_{i1}, l_{i2}, \ldots, l_{iM})$ and l_i are the M different attributes for $i = 1,2,...,X$. The relation U is a very large relation consisting of $M >> N$ attributes. The important concept in the development of this paper is that a query can be defined as $R_t(n(t)) = (R_t(1), R_t(2), \ldots, R_t(n(t)))$ 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

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

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 $A = \{v_1, v_2, \ldots, v_Y\}$ where $v_o = (l_{o1}, l_{o2}, \ldots, l_{oM})$ and l_o are the M different attributes for $o = 1,2, \ldots, Y$. 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.

3.2. *Result Cost Function*

We consider that 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 that the user considers 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 an 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 that measures the result relevance and HW is 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 an N-dimension vector associated with the result and n is the attribute index from the query $R_t(n(t))$:

$$HF[n] = \begin{cases} \frac{N-n}{N} & \text{if } SD[n] > 0 \\ 0 & \text{if } SD[n] = 0 \end{cases} \tag{3}$$

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

We define Score Dimension $SD[n]$ as an 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 on whether 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 that 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 positions:

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

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

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

(*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 higher score on the domain values that the user has filled more 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 value.

3.3. *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 $C_P = \{v_1, v_2, \ldots, v_P\}$. We consider v_P as a vector of M dimensions; $v_p = (l_{p1}, l_{p2}, \ldots, l_{pM})$ where l_p are the M different attributes for $p = 1,2, \ldots, P$. Similarly, the user can also select a subset of Q irrelevant results, C_Q of A, $C_Q = \{v_1, v_2, \ldots, v_Q\}$. We consider v_q as a vector of M dimensions; $v_q = (l_{q1}, l_{q2}, \ldots, l_{qM})$ where lq are the M different attributes for $q = 1,2, \ldots, Q$. Based on the user iteration, our Intelligent Internet Search Assistant provides 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 is the attribute index from the query $R_t(n(t))$ and $SD_p[n]$ is 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:

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

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

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 below:

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

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. An 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 based on the dimensions or attributes the user has introduced on the initial query.

3.4. 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 and therefore the subset of results the user has considered as relevant may have other relevant concepts hidden that the user did not consider in 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 expanded 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 consists 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 of irrelevant results, C_Q.

Let us consider C_P as a set that consists of P M-tuples $C_P = \{v_1, v_2, \ldots, v_P\}$ where v_P is a vector of M dimensions; $v_P = (l_{p1}, l_{p2}, \ldots, l_{pM})$ 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 $DA[m]$:

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

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

where M is the total number of attributes that form the relation U. The correlation vector $\sigma[m]$ 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 $\sigma[m]$:

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

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 $\sigma[m]$ is less than the average correlation 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 no new relevant results to be shown to the user.

3.5. *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.

3.6. *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

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

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 is the result attribute index, M the total number of attributes and s is the iteration number. Our ISA then recalculates the potential of each of the results based on the updated network weights and reorders them, showing to the user the results that have a higher potential or score.

4. Validation

The 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 ISA algorithm with a set of three different experiments. Users in the experiments can both choose between the different web search engines and the N number of results they would like to retrieve from each one. We propose the following formula to measure web search quality; it is based on the concept that a better search engine provides a list of more relevant results in the top positions. In an list of N results, we score N to the first result and 1 to the last result, the value of the quality proposed is then the summation of the position score based of each of the selected results. Our definition of Quality, Q, can be defined as:

$$Q = \sum_{i=1}^{Y} RSE_i \tag{22}$$

where RSE_i is the rank of the result i in a particular search engine with a value of N if the result is in the first position and 1 if the result is the last one. Y is the total number of results selected by the user. The best web search engine would have the largest Quality value. We define normalized quality, $\overline{Q}$, as the division of the quality, Q, by the optimum value, which it is when the user considers relevant all the results provided by the particular search engine. In this situation, Y and N have the same value:

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

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

We define I as the quality improvement between a web search engine and a reference:

$$I = \frac{QW - QR}{QR} \tag{24}$$

where I is the improvement, QW is the quality of the web search engine and QR is the quality reference; we use the Quality of Google as QR in our validation exercise.

4.1. Web Search Engine

In our first experiment we have asked our validators to search for different queries using only Google; ISA provides a set of reordered results from which the user needs to select the relevant results. We show the average values for the 20 different queries, the average number of results retrieved by Google and the average number of results selected by the user. We represent the normalized quality of Google and ISA with the improvement of our algorithm against Google. In our second experiment, ISA provides a reordered list from which the user needs to select which results are relevant. Our ISA reorders the results using the dimension relevant centre point providing the user with another reordered result list, from which the user needs to select the relevant ones. We show the average values for the 16 different queries, the average number of results selected by the user and the average number of results selected. We also represent the normalized quality of Google, ISA and the ISA with the relevant circle iteration including the improvement against Google in both scenarios. In our third experiment, validators can select the web search engine they would like their results to be retrieved from; as in our first experiment, the users need to select the relevant results. Our ISA combines the results retrieved from the different Web search engines selected. We present the average values for the 18 different queries. We show in the normalized quality of each web search engine selected including our ISA; because users can choose any web search engine, we are not introducing the improvement value as we do not have a unique reference web search engine.

Table 1. Web search engine validation.

Experiments 1: 20 queries						
Results retrieved	Results selected	Google Q	ISA Q	ISA I	ISA Circle Q	ISA Circle I
19.35	8.05	0.4626	0.4878	15.39%	-	-
Experiments 2: 16 queries						
Results retrieved	Results selected	Google Q	ISA Q	ISA I	ISA Circle Q	ISA Circle I
21.75	8.75	0.4451	0.4595	18%	0.4953	26%
Experiment 3: 18 Queries						
Web	Google	Yahoo	Ask	Lycos	Bing	ISA
$\overline{Q}$	0.2691	0.2587	0.3454	0.3533	0.3429	0.4448

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

Table 2. ISA learning validation.

Gradient Descent Learning: 17 queries								
Web	ISA	I	Web	ISA	I	Web	ISA	I
0.41	0.58	43%	0.45	0.61	14%	0.46	0.62	8%
Reinforcement Learning: 12 queries								
Web	ISA	I	Web	ISA	I	Web	ISA	I
0.42	0.57	34%	0.47	0.67	36%	0.49	0.68	0.0%

4.2. ISA Learning

Users in the experiments can choose between Google and Bing with either Gradient Descent or Reinforcement Learning. Our ISA then collects the first 50 results from the web search engine selected, reorders them according to its cost function and finally shows to the user the first 20 results. We consider 50 results is a good approximation of search depth as more results can add clutter and irrelevance; 20 results is the average number of results read by a user before he or she launches another search if he does not find any relevant answer. ISA reorders results while learning in the two step iterative process, showing only the best 20 results to the user. We present the average Quality values of the web search engine and ISA for the 29 different queries searched for by different users, the learning type and the web search engine used. The first 'I' represents the improvement from ISA against the web search; the second 'I' is between ISA iterations 2 and 1 and finally the third 'I' is between the ISA iterations 3 and 2.

5. Conclusions

We have proposed a novel approach to web searching 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 Internet Search Assistant generally performs slightly better than Google and other web search engines. However, this evaluation may be biased because users tended to concentrate on the first results provided, which were the ones we showed in our algorithm. Our ISA adapts and learns from a user's previous relevance measurements, increasing significantly its quality and improvement within the first iteration. The Reinforcement Learning algorithm performs better than the Gradient Descent one. Although the Gradient Descent algorithm provides a better quality on the first iteration; the Reinforcement Learning algorithm outperforms on the second one due its higher learning rate. Both have a residual learning on their third iteration. The Gradient Descent algorithm would have been the preferred learning algorithm if only one iteration is required;

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

however, Reinforcement Learning would have been a better option in the case of two iterations. It is not recommended to require three iterations because learning is only residual.

References

1. E. Gelenbe (2010) Search in unknown random environments. *Physical Review E*, **82**, 061112.
2. O.H. Abdelrahman and E. Gelenbe (2013) Time and energy in team-based search. *Physical Review E*, **87**, 032125.
3. E. Gelenbe and O. H. Abdelrahman (2014) Search in the universe of big networks and data. *IEEE Networks*, **28**(4), pp. 20–25.
4. E. Gelenbe (2009) Analysis of single and networked auctions. *ACM Transactions on Internet Technology*, **9**(2).
5. F. Scarselli, S. Liang, M. Hagenbuchner and A. Chung (2005) Adaptive page ranking with neural networks. *Proceeding WWW '05 Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, pp. 936– 937.
6. M. Chau and H. Chen (2007) Incorporating web analysis into neural networks: an example in Hopfield net searching. *IEEE Transactions on Systems and Cybernetics – Part C: Applications and Reviews*, **37**(3), pp. 352–358.
7. S. Bermejo and J. Dalmau (2003) 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*, pp. 711–718.
8. C. Burgues, T. Shaked, E. Renshaw, L. Lazier, M. Deeds, N. Hamilton and G. Hullender (2005) Learning to rank using gradient descent. *ICML '05 Proceedings of the 22nd International Conference on Machine Learning*, pp. 89–96.
9. B. Shu and S. Kak (2009) A neural network-based intelligent metasearch engine. *Information Sciences, Informatics and Computer Science*, **120**, pp. 1–11.
10. J. Boyan, D. Freitag and T. Joachims (1996) A machine learning architecture for optimizing web search engines. *Proceedings of the AAAI Workshop on Internet-based Information Systems*.
11. X. Wang and L. Zhang (2011) Search engine optimization based on algorithm of BP neural networks. *Proceedings of the Seventh International Conference on Computational Intelligence and Security*, pp. 390–394.
12. E. Gelenbe (1989) Random neural network with negative and positive signals and product form solution. *Neural Computation*, **1**, 502–510.
13. E. Gelenbe (1993) Learning in the recurrent Random Neural Network. *Neural Computation*, **5**, 154–164.
14. E. Gelenbe and S. Timotheou (2008) Random neural networks with synchronized interactions. *Neural Computation*, **20**(9), pp. 2308–2324.
15. E. Gelenbe, R. Lent and Z. Xu (2011) Towards networks with cognitive packets *Performance and QoS of Next Generation Networking* (London: Springer), pp. 3–17.
16. E. Gelenbe (2009) Steps towards self-aware networks. *Communications of the ACM*, **52**(7), pp. 66–75.
17. E. Gelenbe and F.J. Wu (2012) Large scale simulation for human evacuation and rescue. *Computers & Mathematics with Applications*, **64**(12), pp. 3869–3880.
18. A. Filippoupolitis, L. Hey, G. Loukas, E. Gelenbe and S. Timotheou (2008) Emergency response simulation using wireless sensor networks.

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press

Proceedings of the First International Conference on Ambient Media and Systems, **21**.
19. E. Gelenbe and T. Koçak (2000) Area-based results for mine detection. *IEEE Transactions on Geoscience and Remote Sensing*, **38**(1), pp. 12–24.
20. C. Cramer, E. Gelenbe and H. Bakircloglu (1996) Low bit-rate video compression with neural networks and temporal subsampling. *Proceedings of the IEEE*, **84**(10), pp. 1529–1543.
21. V. Atalay, E. Gelenbe and N. Yalabik (1992) The random neural network model for texture generation. *International Journal of Pattern Recognition and Artificial Intelligence*, **6**(1), pp. 131–141.

About the Author

Will Serrano is a Chartered Telecoms Systems Designer specializing in technical design and design management of large and high-profile infrastructure projects, from airports to rail stations to energy plants such as CrossRail; Heathrow Airport and New Tube for London. He has helped deliver solutions in Local Area Networks; Wide Area Networks; Wireless LAN; Voice; Cloud and Cybersecurity. Serrano holds a Master's Degree in Telecommunication Systems & Networks. He is a Member of the Institution of Engineering and Technology. He is a PhD researcher with the Intelligent Systems and Networks Department at Imperial College London, UK.

https://doi.org/10.1017/S1062798716000594 Published online by Cambridge University Press