SINGLE DOCUMENT TEXT SUMMARIZATION USING RANDOM INDEXING AND NEURAL NETWORKS

Niladri Chatterjee and Avikant Bhardwaj
Department of Mathematics, Indian Institute of Technology Delhi, New Delhi, India

Keywords: Text summarization, Neural networks, Random indexing, Word space modelling.

Abstract: This paper presents a new extraction-based summarization technique developed using neural networks and Random Indexing. The technique exploits the advantages that a neural network provides in terms of compatibility and adaptability of a system as per the user. A neural network is made to learn the important properties of sentences that should be included in the summary through training. The trained neural network is then used as a sieve to filter out the sentences relevant for corresponding summary. Neural network along with Random Indexing extracts the semantic similarity between sentences in order to remove redundancy from the text to great success. One major advantage of the proposed scheme is that it takes care of human subjectivity as well.

1 INTRODUCTION

Automatic text summarization has become an important tool for interpreting text information, due to the extension of Internet and the abundance of knowledge in textual form available on the World Wide Web. However, it provides more information than is usually needed. Hence, extracting a large quantity of relevant information has put it in focus for researchers in NLP (Kaikhah, 2004).

The prime focus of the present work is to generate an extractive summary of a single document using neural networks. The basic difficulty seen in summarization is the subjectivity of generated summary in view of different users, and almost all the current techniques tend to ignore to incorporate this feature. However, our intuition is that using Neural Networks (NN) one can generate summaries much closer to human extracted summaries as they are trained on already available standard human summaries to produce a better result. In this work we have combined NN and Random Indexing (Sahlgren, 2005) which have provided significant improvement over the commercially available summarizing tools.

Traditional extractive summarization techniques are typically based on simple heuristic features of the sentences. Though there has been a considerable and thorough research work on graph based or other implementations of text summarizer, there has not been much work using Artificial Intelligence techniques in general, and neural networks, in particular, except for some heuristic approaches[1]. Researchers have tried to integrate machine learning techniques into summarization with various features, such as sentence length cut-off, fixed-phrase, thematic word, and many more (Kupiec et al., 1995). Apart from those, some commercially available extractive summarizers like Copernic[2] and Microsoft Office Word summarizer [3] use certain statistical algorithms to create a list of important concepts and hence generate a summary.

This paper is organized as follows. Section 2 describes neural networks along with back propagation algorithm. Section 3 explains the technique of Random Indexing. Sections 4 and 5 discuss the experimental set ups and the results obtained, respectively. In Section 6 we conclude the paper.

2 NEURAL NETWORKS

An Artificial Neural Network (ANN) is a mathematical model or computational model that tries to simulate the structure and/or functional aspects of biological neural networks. Neural networks are non-linear statistical data modeling tools & are used to model complex relationships between inputs and outputs and to find patterns in data (Rojas, 1996). The artificial

[1]See (Luhn, 1958; Edmundson, 1969; Mani and Bloedorn, 1999; Kaikhah, 2004)
[2]*www.copernic.com/en/products/summarizer/*
[3]*www.microsoft.com/education/autosummarize.mspx/*

neurons, constitutive units in an ANN, are mathematical function observed as a rudimentary model, or abstraction of biological neurons. Mathematically, let there be $n+1$ inputs with signals x_0 to x_n and weights w_0 to w_n, respectively. Usually, the x_0 input is assigned the value $+1$, which makes it a bias input with $w_0 = b$. This leaves only n actual inputs to the neuron: from x_1 to x_n. The output of such neuron is (where φ is the activation function):

$$y = \varphi\left(\sum_{j=0}^{n} w_j x_j\right) \tag{1}$$

In the “learning” phase of a neural network, we try to find best approximations of the different weights $w_0, w_1, \ldots, w_n$. This is done by minimizing a cost function which gives a measure of the distance between a particular solution and the optimal solution that we try to achieve. Numerous algorithms are available for training neural network models (Bishop, 2005); most of them can be viewed as a straightforward application of optimization theory and statistical estimation. We have implemented one of the more popular learning algorithms called Backpropagation algorithm. It is supervised learning in an iterative way, where the error produced in each iteration is used to improve the weights corresponding to each input variable and thus forcing the output value to converge to the known value.[4]

In order to use neural networks for our approach, we first require that the sentences are in some mathematical model so that we can use them as input in our network. For that purpose, we introduce Word Space Modelling, which is a spatial representation of word meaning, through Random Indexing (RI) (Chatterjee and Mohan, 2007). RI transforms every sentence into a vector location in word space and NN then uses that vector as input for computational purposes.

3 WORD SPACE MODEL

The Word-Space Model (Sahlgren, 2006) is a spatial representation of word meaning. It associates a vector with each word defining its meaning. However, the Word Space Model is based entirely on language data available. When meanings change, disappear or appear in the data at hand, the model changes accordingly. The primary problem with this representation is that we have no control over the dimension of the vectors. Consequently, use of such a representation scheme in NN-based model lacks appropriateness. We use a Random Indexing based representation scheme to deal with this problem.

[4]See (Rojas, 1996) and (Bishop, 2005) for details of Backpropagation algorithm.

3.1 Random Indexing Technique

The Random Indexing was developed to tackle the problem of high dimensionality in Word Space model. It removes the need for the huge co-occurrence matrix by incrementally accumulating context vectors, which can then, if needed, be assembled into a co-occurrence matrix (Kanerva, 1988).

In Random Indexing each word in the text is assigned a unique and randomly generated vector called the *index vector*. All the index vectors are of the same predefined dimension R, where R is typically a large number, but much smaller than n, the number of words in the document. The index vectors are generally sparse and ternary i.e. they are made of three values chosen from $\{0, 1, -1\}$, and most of the values are 0. When the entire data has been processed, the R-dimensional context vectors are effectively the sum of the words' contexts. For illustration we can take the example of the sentence

A beautiful saying, a person is beautiful when he thinks beautiful.

Let, for illustration, the dimension R of the index vector be 10. The context is defined as one preceding and one succeeding word. Let ‘person’ be assigned a random index vector: $[0,0,0,1,0,0,0,0,-1,0]$ and ‘beautiful’ be assigned a random index vector: $[0,1,0,0,-1,0,0,0,0,0]$. Then to compute the context vector of ‘is’ we need to sum up the index vector of its context which is, $[0,1,0,1,-1,0,0,0,-1,0]$. The space spanned by the context vectors can be represented by a matrix of order $W \times R$, where i^{th} row is the context vector of i^{th} distinct word.

If a co-occurrence matrix has to be constructed, R-dimensional context vectors can be collected into a matrix of order $W \times R$, where W is the number of unique word types, and R is the chosen dimensionality for each word. Note that this is similar to constructing an n-dimensional unary context vector which has a single 1 in different positions for different words and n is the number of distinct words. Mathematically, these n-dimensional unary vectors are orthogonal, whereas the R-dimensional random index vectors are nearly orthogonal. However, most often this does not stand on the way of effective computation. On the contrary, this small compromise gives us huge computational advantage as explained below. There are many more nearly orthogonal than truly orthogonal directions in a high-dimensional space (Sahlgren, 2005). Choosing Random Indexing is an advantageous trade-off between the number of dimensions and orthogonality, as the *R*-dimensional random index vectors can be seen as approximations of the *n*-dimensional unary vectors.

3.2 Assigning Semantic Vectors to Documents

The average term vector can be considered as the central theme of the document and is computed as:

$$\vec{x}_{mean} = \frac{1}{n}\sum_{i=0}^{n}\vec{x}_i \tag{2}$$

where *n* is the number of distinct words in the document. While we compute the semantic vectors for the sentences we subtract $\vec{x}_{mean}$ from the context vectors of the words of the sentence to remove the bias from the system (Higgins et al., 2004). The semantic vector of a sentence is thus computed as:

$$\vec{x}_{semantic} = \frac{1}{m}\sum_{i=0}^{m}\left(\vec{x}_i - \vec{(x)}_{mean}\right) \tag{3}$$

where *m* is the number of words in the focus sentence and x_i refer to the context vector of i_{th} word. Note that subtracting the mean vector reduces the magnitude of those term vectors which are close in direction to the mean vector, and increases the magnitude of term vectors which are most nearly opposite in direction from the mean vector. Thus the words which occur very commonly in a text, such as the auxiliary verbs and articles, will have little influence on the sentence vector so produced. Further, the terms whose distribution is most distinctive will be given the maximum weight. The semantic vector of sentence thus obtained is fed into NN as input vector, and the corresponding output from NN is ranking of the sentence, which is a real number between 0 and 1.

4 EXPERIMENTAL SETUP

Our experimental data set consists of 25 documents containing 300 to 700 words each. The processing of each document to generate a summary has been carried out as follows:

4.1 Mapping of Words on Word Space Model

We have implemented the mapping of words onto the word space by three methods, namely *Narrow Window Approach*, *Extended Window Approach* and *Sentence Context Approach*. In ‘narrow window’ approach, each word in the document was initially assigned a unique randomly generated index vector of the dimension 100 with ternary values $(1,-1,0)$. The index vectors were so constructed that each vector of 100 units contained two randomly placed 1 and two randomly placed -1s, rest of the units were assigned 0 values. Each word was also assigned an initially empty context vector of dimension 100. We defined the context by a 2×2 sliding window on the focus word. The context of a given word was also restricted in one sentence, i.e. across sentence windows were not considered. Experiments conducted at SICS, Sweden (Karlgren and Sahlgren, 2001) have indicate that a 2×2 window is preferable for acquiring semantic information. Once context vectors for words are generated, we then generate semantic vectors for individual sentences as described earlier. The second approach with ‘extended window’, considers a sliding window of size 4×4 instead of 2×2. Thus the problem with the kind of words which have same immediate context words but different meanings is resolved by taking larger window. For example consider the two sentences, *Doing good is humane* and *Doing bad is inhumane*. Here, *good* and *bad* have same immediate context words, but have different meanings. However if we extend the window for context words, then we will have much better approximation of their meaning. In the third approach of ‘sentence context’, we have updated the input vectors not from the semantic vectors of sentences, but by realizing the semantic vectors as context vectors for sentences, and hence create the input vector by adding the 4×4 context window’s context vectors.

We now have the semantic vectors of the sentences of the document which act as the input pattern vectors for our neural network. The sentence is selected or rejected on the basis of the output given by the network for the corresponding semantic vector.

4.2 Text Summarization Process

The proposed approach summarizes given text documents through a two phase process, the details of which are discussed below:

1. (One time) Training of the Neural Network.
2. Generating summary of a user defined text.

The first step involves one time training of a neural network through change in its weights to recognize the type of sentences that should be included in the summary. Once training is done, any number of single document text files can be summarized in the second step, which uses the modified neural network from first step to sieve the text to select only the highly ranked sentences and create the required summary.

4.2.1 Training of the Neural Network

The first phase of the summarization process involves training the neural networks to learn the types of sentences that should be included in the summary. This is accomplished by training the network with sentences in several test paragraphs where each sentence is identified as to whether it should be included in the summary or not. For this purpose the training texts and the corresponding summaries are provided by the user all in separate text files after appropriate pre-processing, viz. formatting the text such that every sentence starts from a new line and there are no blank lines in between etc. The neural network learns the patterns inherent in sentences that should not be included in the summary and those that should be included. We use a three layered Feed-Forward neural network. It has been proven to be a universal function approximator, which is considered to be very efficient in discovering patterns and approximating the inherent function.

Fig. 1 shows the structure of the neural network used in this work. As shown, the network contains three layers- Input Layer, one Hidden Layer and the Output Layer. The output layer of our network contains only one neuron whose output gives the rank of a sentence on the basis of which the sentence is selected or rejected. However, the no. of neurons in the input layer of our network is 101 (one for the bias and the remaining for the 100 inputs each corresponding to one dimension of the index vector) and the no. of neurons in the hidden layer is 65. The no. of neurons in the hidden layer has been selected arbitrarily. However, that can also be calculated using the following formula[5]:

$$numHid = \frac{numInput}{2} + \sqrt{numPat} \qquad (4)$$

where *numHid* = No. of Hidden layer neurons, *numInput* = No. of inputs including bias and *numPat* = No. of patterns to be trained. The number of hidden layer neurons should not be, in any case, less than the number obtained from above formula, as it leads to inconsistency in weights updating of larger texts. However, more number of neurons will result in slower training. Hence in order to strike a balance between training time and consistency, we decided to have 65 neurons in the hidden layer. In our case, the training patterns are the semantic vectors of the sentences of the training documents. The overall error is calculated using the mean square error of individual error generated by each training pattern. The maximum number of iterations kept for the algorithm is 10000 and the tolerance limit for the total error has

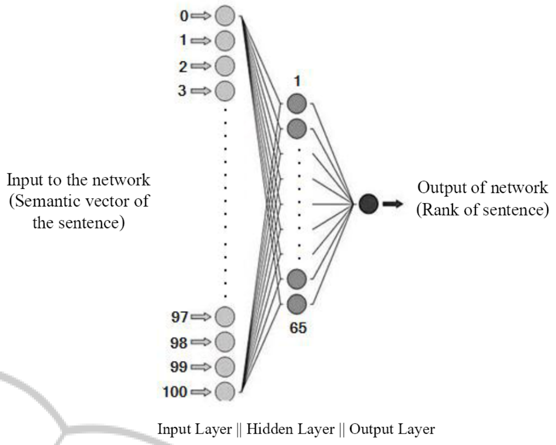
0
1
2
3
97
98
99
100
1
65
Input to the network (Semantic vector of the sentence)
Output of network (Rank of sentence)
Input Layer || Hidden Layer || Output Layer

Figure 1: A schematic diagram of the neural network.

been kept as 0.0000003, which have been found out to be optimal by repeated trials. The learning rates for changing the weights between both input-output layers and hidden-output layers are 0.01. The activation function used for the hidden layer's neurons is the sigmoid function and for output layer's neuron it is linear function.

4.2.2 Generating Summary of an Arbitrary Text

Once the network has been trained, it can be used as a tool to filter sentences in any paragraph and determine whether each sentence should be included in the summary or not. The program finds the semantic vectors of the sentences of given document. Using the weight values found in the training step the outputs of all these semantic vectors are calculated. This output of a semantic vector corresponds to the rank of the corresponding sentence. The rank of a sentence is directly proportional to its priority/importance within the document. The sentences are then sorted using modified quick-sort technique. On the basis of their ranks the high priority sentences are selected to create the summary. Please note that in the present case we are selecting summaries based on the percentage of total number of words in the document. Since in terms of sentences it may not give an exact number, we have chosen top few sentences according to their ranks, until the total word count is not exceeding the percentage mentioned along with a leeway of further 10%. For illustration, for 25% summaries, we selected top ranked sentences until the total sum is not exceeding 27.5% of total no. of words in the text.

[5] *http://www.wardsystems.com/manuals/ neuroshell2*

Table 1: Precision, recall and F values for tested files (rounded off to second decimal).

Text No.		**Narrow Window**			**Extended Window**			**Sentence Context**			**Copernic**			**MS Word**		
(Size of text)		p	r	F	p	r	F	p	r	F	p	r	F	p	r	F
1	25%	0.50	0.50	0.50	0.50	0.75	0.60	**0.75**	**0.80**	**0.77**	0.33	0.60	0.43	0.50	0.50	0.50
(359)	50%	0.38	0.50	0.43	0.43	0.50	0.47	**0.50**	**0.67**	**0.57**	0.43	0.45	0.44	0.25	0.33	0.29
2	25%	1.00	0.50	0.67	**0.83**	**0.70**	**0.76**	0.41	0.44	0.42	0.50	0.67	0.57	0.10	0.50	0.17
(298)	50%	0.50	0.38	0.43	0.55	0.63	0.59	**0.63**	**0.63**	**0.63**	0.50	0.33	0.38	0.50	0.38	0.43
3	25%	0.00	0.00	NaN	**0.78**	**0.78**	**0.78**	0.60	0.67	0.63	0.43	0.60	0.50	0.67	0.50	0.57
(396)	50%	0.50	0.55	0.53	**0.70**	**0.78**	**0.74**	0.67	0.67	0.67	0.57	0.63	0.60	0.33	0.44	0.38
4	25%	0.10	0.40	0.16	0.33	0.30	0.31	**1.00**	**0.89**	**0.93**	0.25	0.50	0.33	0.00	0.00	NaN
(408)	50%	0.38	0.33	0.35	0.63	0.55	0.59	**0.89**	**0.78**	**0.83**	0.33	0.55	0.41	0.25	0.22	0.24
5	25%	0.25	0.25	0.25	0.20	0.40	0.27	**0.80**	**0.67**	**0.73**	0.63	0.50	0.56	0.40	0.50	0.44
(519)	50%	0.50	0.60	0.54	0.50	0.60	0.54	**0.50**	**0.60**	**0.54**	0.40	0.55	0.46	0.50	0.60	0.54
6	25%	0.40	0.50	0.44	0.70	0.89	0.78	0.67	0.33	0.44	0.40	0.60	0.48	**1.00**	**1.00**	**1.00**
(504)	50%	0.55	0.45	0.50	0.63	0.55	0.58	0.67	0.63	0.64	0.66	0.75	0.70	**0.83**	**0.78**	**0.80**
7	25%	0.55	0.67	0.59	0.80	0.67	0.73	**0.83**	**1.00**	**0.91**	0.50	0.43	0.46	0.50	0.43	0.46
(694)	50%	0.46	0.71	0.56	0.40	0.57	0.47	0.43	0.71	0.54	**0.62**	**0.57**	**0.59**	0.30	0.43	0.35
8	25%	0.33	0.60	0.43	0.50	0.50	0.50	**0.75**	**0.60**	**0.67**	0.55	0.55	0.55	0.20	0.33	0.25
(681)	50%	0.30	0.36	0.33	0.36	0.46	0.40	**0.80**	**0.74**	**0.77**	0.40	0.66	0.50	0.53	0.73	0.62
9	25%	**0.67**	**1.00**	**0.80**	0.25	0.50	0.33	0.20	0.33	0.25	0.50	0.50	0.50	0.40	0.67	0.50
(608)	50%	0.40	0.46	0.43	0.44	0.61	0.52	0.50	0.69	0.58	**0.80**	**0.75**	**0.77**	0.60	0.69	0.64
10	25%	0.33	0.33	0.33	0.67	0.67	0.67	**1.00**	**0.75**	**0.86**	0.67	0.67	0.67	0.67	0.50	0.57
(388)	50%	0.33	0.33	0.33	0.55	0.55	0.55	**0.89**	**0.80**	**0.84**	0.67	0.50	0.57	0.55	0.55	0.55
Average	25%	0.41	0.48	0.43	0.56	0.62	0.57	**0.70**	**0.65**	**0.66**	0.48	0.56	0.51	0.44	0.50	0.54
	50%	0.43	0.47	0.44	0.52	0.58	0.55	**0.65**	**0.69**	**0.66**	0.54	0.57	0.54	0.46	0.52	0.48

5 RESULTS

We have first trained our summarizer on 15 different texts and their standard summary provided by DUC. Each document of DUC 2002 corpus is accompanied by two different abstracts manually created by professional abstractors. For each abstract so created, we made a corresponding extract summary, by replacing restructured sentences with the closest sentence(s) from the original document. Then we have taken their union to have a reference summary (S_{ref}). For evaluation, our results have been compared with the reference summary thus created. We ran our experiments on a different set of 10 texts and computed extracts at 25% and 50% levels. We then compare our candidate summary, one from each approach, (denoted by S_{cand}) with the reference summary and compute the precision (p), recall (r) and F values (Yates and Neto, 1999). We also compute the p, r and F values for the summaries generated by Copernic and MS Word summarizers. The values obtained for 10 test documents only (due to space constraints) have been shown in Table 1.

Table 1 shows the comparison of the 25% and 50% summaries of 10 randomly selected text files of DUC 2002, outside the training texts. For each of them we computed the p, r and F values for all the three approaches proposed by us, and also for Copernic and MS Word summarizers. Out of the 20 cases, Sentence Context approach gave the best results in as many as 13 cases. This was followed by the Extended Window approach, which gives the best result in 3, while MS Word, Copernic and Narrow Window approach gave the best result only in 2, 1 and 1 cases, respectively. For both 25% and 50% summaries the average F-values for the Sentence Context approach is 0.66. The next highest average F-values for 25% summaries is 0.57 for the Extended Approach, whereas it is 0.56 for the 50% summaries again for the Extended Window approach.

The results of these limited experiments clearly indicate that the summaries created by the proposed scheme (in particular for the Sentence Context and Extended Window approach) are very close to the human generated summaries, compared to the existing summarizers at both 25% and 50% level. The results are certainly very promising. However, it is too early to predict how the neural network along with Random Indexing based scheme will work in general. A lot more experiments need to be done to come to a conclusion. We are currently working towards this direction.

6 CONCLUSIONS

In this work we propose a scheme for text summarization using Random Indexing and neural networks. The approach exploits the similarity in the meaning of the words by mapping them onto the word space and removing less important sentences by sieving them through a neural network already trained on a set of summarized text.

The problem of high dimensionality of the semantic space has been tackled by employing Random Indexing which is less costly in computation and memory consumption compared to other dimensionality reduction approaches. The selection of features as well as the selection of summary sentences by the human reader from the training paragraphs plays an important role in the performance of the network. The network is trained according to the style of the human reader; and also recognizing to which sentences in a paragraph the human reader puts more emphasis. This, in fact, is an advantage that the proposed approach provides. This allows an individual reader to train the neural network according to one's own style. Furthermore, the selected features can be modified to reflect the reader's needs and requirement. In future we plan to create more complex neural network structure, involving better activation functions, so as to smooth out some abruptness that we encountered in the current schemes. Moreover, in our present evaluation we have used measures like precision, recall and F which are used primarily in the context of information retrieval. In future we intend to use more summarization-specific techniques to measure the efficacy of our scheme.

In order to develop the proposed technique into an efficient summarization tool we need to answer quite a few questions viz. what is the right value for R, the dimension of the index vectors; what is the right number of nodes in the neural network etc. We are aiming at finding optimum values for these parameters for the proposed model.

In the present scheme we have used three different approaches: Narrow Window, Extended Window and Sentence Context. We have noticed that the on the average the *Sentence Context* approach produces the best result. But it is not uniformly best in several cases other approaches have produced results better than this approach. We need to do more experimentation with the Sentence Context approach in order to elevate its performance level. Alternatively, we may have to suitably combine the three proposed approaches in order to produce good results uniformly. We are currently working towards these goals.

REFERENCES

Bishop, C. M. (2005). *Neural Networks for Pattern Recognition*. Oxford Press.

Chatterjee, N. and Mohan, S. (2007). Extraction-based single-document summarization using random indexing. In *ICTAI (2)*, pages 448–455.

Edmundson, H. P. (1969). New methods in automatic extracting. *J. ACM*, 16(2):264–285.

Higgins, D., Burstein, J., Marcu, D., and Gentile, C. (2004). Evaluating multiple aspects of coherence in student essays. In *HLT-NAACL*, pages 185–192.

Kaikhah, K. (2004). Text summarization using neural networks. *WSEAS Transactions on Systems3*, 3:960–963.

Kanerva, P. (1988). *Sparse distributed memory*. Cambridge: MIT Press.

Karlgren, J. and Sahlgren, M. (2001). *From words to understanding: Foundations of real-world intelligence*. CSLI Publications.

Kupiec, J., Pedersen, J. O., and Chen, F. (1995). A trainable document summarizer. In *SIGIR*, pages 68–73.

Luhn, H. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development*.

Mani, I. and Bloedorn, E. (1999). Summarizing similarities and differences among related documents. *Inf. Retr.*, 1(1-2):35–67.

Rojas, R. (1996). *Neural Networks: A Systematic Introduction*. Springer-Verlag, Berlin.

Sahlgren, M. (2005). An introduction to random indexing. In *Proceedings of the Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering*.

Sahlgren, M. (2006). *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. dissertation, Department of Linguistics, Stockholm University.

Yates, R. and Neto, B. (1999). *Modern Information Retrieval*. Pearson Education.