Summarising text with a genetic algorithm-based sentence extraction

Vahed Qazvinian*

Department of Computer Engineering,
Sharif University of Technology,
Tehran, Iran
E-mail: qazvinian@ce.sharif.edu
*Corresponding author

Leila Sharif Hassanabadi

Department of Computer Sciences,
Shahid Beheshti University,
Tehran, Iran
E-mail: l_sharif@sbu.ac.ir

Ramin Halavati

Department of Computer Engineering,
Sharif University of Technology,
Tehran, Iran
E-mail: halavati@ce.sharif.edu

Abstract: Automatic text summarisation has long been studied and used. The growth in the amount of information on the web results in more demands for automatic methods for text summarisation. Designing a system to produce human-quality summaries is difficult and therefore, many researchers have focused on sentence or paragraph extraction, which is a kind of summarisation. In this paper, we introduce a new method to make such extracts. Genetic Algorithm (GA)-based sentence selection is used to make a summary, and once the summary is created, it is evaluated using a fitness function. The fitness function is based on three following factors: Readability Factor (RF), Cohesion Factor (CF) and Topic-Relation Factor (TRF). In this paper, we introduce these factors and discuss the Genetic Algorithm with the specific fitness function. Evaluation results are also shown and discussed in the paper.

Keywords: automatic text summarisation; sentence extraction; genetic algorithm.

Reference to this paper should be made as follows: Qazvinian, V., Hassanabadi, L.S. and Halavati, R. (2008) 'Summarising text with a genetic algorithm-based sentence extraction', *Int. J. Knowledge Management Studies*, Vol. 2, No. 4, pp.426–444.

Biographical notes: Vahed Qazvinian recieved his BSc in Computer Engineering (Software Discipline) from Sharif University of Technology, Tehran, Iran in 2007. As of September 2007, he has been pursuing his graduate studies in the School of Information, University of Michigan, Ann Arbor. His current research is on information retrieval and natural language processing. More specifically, he is currently working on 'Scientific Paper Summarisation'.

Leila Sharif Hassan abadi received his MS and PhD in Computer Science from Saitama University, Saitama, Japan, in 2000 and 2003, respectively. She is an Assistant Professor of Computer Science at Shahid Behehti University, Tehran, Iran. Her research interests include soft-computing, information retrieval, intelligent control and industrial robotic.

Ramin Halavati received his MS in Artificial Intelligence from Sharif University of Technology, Tehran, Iran in 2003, and is now a PhD candidate of Artificial Intelligence in Computer Engineering Department of Sharif University of Technology. His current research is on pattern recognition by generating hierarchies of sub-patterns and super-patterns that can lead to faster or more accurate recognition of patterns of increasing of pattern learning speed by gaining more experiences from previous tasks. He has been the author of about 40 papers by now and is supposed to be graduated in 2008.

1 Introduction

One approach to respond to the rapid growth of information is to use text summarisation for faster and more efficient processing by human as well as computer agents. A human performs following steps to do a summarisation (Mitra et al., 1997). Firstly, she understands the content of the document, secondly, identifies the most important pieces of information in the text, and finally, writes up this information. Automating the first and last part is beyond the state of the art for unconstrained documents (Brandow et al., 1995). So, summarisation which is based on semantics would be a future success, but for now, making summaries reduces to the task of Extraction. Techniques for sentence extraction have been widely proposed (Kupiec et al., 1995; Mitra et al., 1997).

Researches on *sentence* or *paragraph extraction* (Mitra et al., 1997; Salton et al., 1996), *trainable summariser* (Kupiec et al., 1995), and *Attribute selection-based* approaches (Goldstein et al., 1999; Silla et al., 2004) have been done and evaluation results are available. Basically summary building depends on the type of analysis on the text (Barzilay and Elhadad, 1997).

In early and classic summarisation systems, the extract was created according to the most frequent words in the text. Using a frequency table (Luhn, 1958), the summary was extracted. Later, linguistic features got involved to help extracting important sentences. Phrases such as 'In Summary', 'In Conclusion', 'This letter describes…' and similar phrases were used to identify important parts of a text (Edmunson, 1969).

Nowadays, researchers are also concerned with the location of sentences. Sentences are considered to be divided in three groups, depending on their location in the paragraph: paragraph-initial, paragraph-final or paragraph-medial (Kupiec et al., 1995). Headings, bold texts, and initial sentences are considered to be important (Edmunson, 1969; Hovy and Lin, 1997).

Some approaches consider sentences separate, then the similarity of each sentence is measured. The summary consists of sentences with the highest salience. There are also methods that adapt the naïve probability model, to find the good summary with some predefined features (Kupiec et al., 1995).

Documents which are summarised with these techniques may lack some features such as readability. A good summary for a document should be readable, as long as it is related to the topic and is a coherent document itself. Creating a summary based on sentence extraction

is a tradeoff among these factors, that is, trying to increase the readability may decrease similarity to topic and vice versa. In the rest of this paper, we are going to clarify these factors and introduce a model for each of them. First in Section 2, a model for representing a document, based on graph theory, is introduced. This presentation will help us later to define the factors in Section 3. Topic Relation Factor (TRF) is first introduced, next, Cohesion Factor (CF), and at last the Readability Factor (RF) is discussed. A Genetic Algorithm (GA) is used to extract the sentences to form a summary; this GA is based on a fitness function, formed by these three factors. The use of GA may prevent finding local maxima in the pathway to the goal, which will likely occur in local search techniques. Polynomial time search cannot be done, unless readability is considered. We will discuss this thoroughly in Section 4. In Section 5, the application of the proposed algorithm on a sample test collection and evaluation results are discussed. Finally, a conclusion is given in Section 6.

2 Document representation

Representing documents and texts as graphs is an old idea and has been widely used (Mitra et al., 1997). This is because of the powerful and effective features we might get from graph theory, and adoption of the problem to a graph-based problem. We use a special kind of graph called 'DAG'. A DAG is a Directed Acyclic Graph, both of whose features – having directions and being acyclic – are useful in our method in this paper.

2.1 Representation

Here, we show how to represent a document using a DAG. The document is divided into sentences and each sentence is considered to be a vertex (or node) of this graph. The weights of edges show the similarity of sentences. The representing graph is made up of two sets, (V, E) where V is the set of vertices and E is the set of edges. To make the graph for a sample document, one should do the following steps:

- $\forall s_i \in Document$ **Add** s_i to V
- $\forall s_i, s_j \in V$, If s_i is before s_j in the document, regarding chronological order, **Then Add** (s_i, s_j) to E.

With the steps above, the graph is made and once it is built, weights should be assigned to its edges.

2.2 Weighting

After introducing the weighting system, we will show how to weight the constructed graph.

2.2.1 Weighting system

The classic $tf\text{-}idf$ (Baeza-Yates and Ribeiro-Neto, 1999) weighting system is used with some adoptions which is similar to IR classical vector model. $tf\text{-}isf$ weights are computed for each sentence, where s_j shows the jth sentence and k_i is ith index term,

$$tf_{i,j} = \frac{\text{freq}_{i,j}}{\max_l \text{freq}_{l,j}}$$

$$isf_i = \log \frac{N}{n_i}$$

$tf_{i,j}$ is said to be 'term frequency' of ith index term in the jth sentence, and isf_i is 'inverse sentence frequency' of ith index term, where N is the number of all sentences and n_i is the number of sentences which contain k_i.

The corresponding weight is therefore computed as,

$$w_{i,j} = tf_{i,j} \times isf_i$$

A vector is created for each sentence and the title. The title plays the role of a query. So, the vector for title is made up of weights below,

$$w_{i,q} = \left(0.5 + \frac{0.5 \times \text{freq}_{i,q}}{\max_l \ \text{freq}_{l,q}}\right) isf_i$$

where $\text{freq}_{i,q}$ is the raw frequency of the term k_i in the text of the information request q (Baeza-Yates and Ribeiro-Neto, 1999). The similarity of each sentence to the title can then be easily computed using *cosine measure*:

$$\text{sim}(s_j, q) = \frac{\overrightarrow{d_j} \cdot \overrightarrow{q}}{|\overrightarrow{d_j}| \times |\overrightarrow{q}|}$$

$$\text{sim}(s_j, q) = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

Similarly, the similarity of two sentences is computed as :

$$\text{sim}(s_m, s_n) = \frac{\sum_{i=1}^{t} w_{i,m} \times w_{i,n}}{\sqrt{\sum_{i=1}^{t} w_{i,m}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,n}^2}}$$

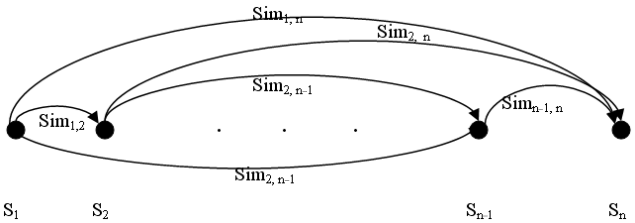
2.2.2 *Weighting the graph*

As depicted in Figure 1 assigning weights to the graph is done as:

- $\forall (s_i, s_j) \in E, W(s_i, s_j) = \text{sim}(s_i, s_j)$.

In fact, the weight of an edge, connecting two vertices, is the similarity of the corresponding sentences.

Figure 1 Weighted DAG representation for a document



At this point, we have represented a document of sentences with a weighted DAG. It should be noted that the edges in the graph are from sentences to ones which are chronologically after them in the text. For instance, if s_i, s_j are two sentences in a document, where s_i is chronologically before s_j, the representing graph will contain the edge (s_i, s_j) but not (s_j, s_i).

3 'Good summary' factors

Here, we discuss the summary and the features it should have. A good summary is a readable summary, the sentences in the summary discuss same things and are related to each other. At the same time, they are related and are similar to the topic, and discuss the same information given by the document topic. Here, we introduce how to measure these metrics. In all factors, it is assumed that summary length is fixed, and factors are computed for that summary length. So from now on, summary length (number of sentences in summary) is supposed to be S, and number of sentences in the main document is supposed to be N.

3.1 Topic Relation factor

A good extract contains sentences that are similar to the text title (Salton and Buckley, 1998; Silla et al., 2004).

Now that the similarities are computed, we can define the TRF. A simple method is to consider the average similarity of sentences in the summary, divided by the maximum average. Suppose, TR is the average similarity to title in a summary (s),

$$\mathrm{TR}_s = \frac{\sum_{s_j \in \text{ summary}} \mathrm{sim}(s_j, q)}{S}$$

Using TR, we compute TRF like this:

$$\mathrm{TRF}_s = \frac{\mathrm{TR}}{\max_{\forall \text{ summary}}(\mathrm{TR})}$$

where max is computed among all possible summaries of length S. To find the max, we should simply average top greater S similarities of all sentences with the topic.

TRF shows the similarity of the created summary to the document title. In summaries where sentences are closely related to the title, TRF is close to 1. But in summaries which are constructed by the sentences far from the title, TRF tends to zero.

3.2 Cohesion factor

CF is a metric to determine whether sentences in the summary talk about the same information or not. In order to know which sentences are useful to make the extract, one should know the relation between sentences, and discover how sentences are related to each other Mitra et al. (1997). A good summary may include sentences which are tightly coupled.

For this purpose, we need the similarity of every pair of sentences. This similarity is computed and accessible by the graph representation. The graph itself can be shown by a matrix. *similarity matrix*[1] is the adjacent matrix of the graph:

$$\begin{pmatrix} \mathrm{Sim}_{1,1} & \mathrm{Sim}_{1,2} & \dots & \mathrm{Sim}_{1,N} \\ \mathrm{Sim}_{2,1} & \mathrm{Sim}_{2,2} & \dots & \mathrm{Sim}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Sim}_{N,1} & \mathrm{Sim}_{N,2} & \dots & \mathrm{Sim}_{N,N} \end{pmatrix}$$

Knowing that there is no edge from proceeding sentences to previous ones, and there is no self-edge, the similarity of every sentence to itself is considered to be 0, and so is the similarity of sentences to those which are chronologically before them. These two assumptions are sorted here:

- $\forall i < N : \text{sim}(s_i, s_i) = 0$
- $\forall i, j < N :$ If(s_i is chronologically before s_j in original document)
  Then $\text{sim}(s_j, s_i) = 0$

N is the total number of sentences in the document. Now we try to form a factor showing the cohesion of sentences. Consider a summary subgraph. The ideal factor could be the average of the weights of all edges in the subgraph, divided by maximum average among all possible summaries. But finding such a maximum (A subgraph with S vertices, whose average weight of all edges is maximum) cannot be solved in polynomial time. In fact, it can be easily shown that this problem is reducible to *max weight-clique* problem. So, we use another approach for CF:

Let C be the average of similarities of all sentences in a summary, it is clear that, C is the average of weights of all edges in the summary subgraph(s):

$$C_s = \frac{\sum_{\forall s_i, s_j \in \text{ summary subgraph}} W\left(s_i, s_j\right)}{N_s}$$

where N_s is the total number of edges in summary subgraph. N_s can easily be computed. Suppose summary nodes are $s_{s_1}, s_{s_2}, \ldots, s_{s_S}$, and S is the total number of sentences in the summary. Then, N_s is number of edges from s_{s_1} to s_{s_j}, $1 < j \leq S$, plus number of edges from s_{s_2} to s_{s_j}, $2 < j \leq S, \ldots$

So,

$$N_s = (S-1) + (S-2) + \cdots = \frac{(S) \times (S-1)}{2}$$

CF should show how summary sentences are close in total, and is defined as below:

$$\text{CF}_s = \frac{\log(C \times 9 + 1)}{\log(M \times 9 + 1)}$$

M is the maximum weight in the graph, that is, M is the maximum similarity of sentences.

$$M = \max_{i,j \leq N} \text{Sim}_{i,j}$$

As it is clear, $C \leq M$, because the maximum edge is whether included in the summary, or not. In both cases, the average of weights in summary is lower than the maximum edge, M.

$$0 \leq C \leq M \Rightarrow 0 \leq CF \leq 1$$

The base of the above logarithms is 10, so
$0 \leq C \leq 1$
$\Rightarrow 1 \leq 9 \times C + 1 \leq 10$
$\Rightarrow 0 \leq \log(9 \times C + 1) \leq 1$
Similarly, $0 \leq \log(9 \times M + 1) \leq 1$.

This formula is reached by some tuning. After several experiments, a logarithmic function is used. Logarithm will help to avoid very low magnitudes of CF when the average is much smaller than the maximum. This is done through changing the division to a logarithmic domain.

If most of the sentences in the extract talk about same topic, CF grows and on the other hand, if they are mostly far from each other, CF tends to 0.

3.3 Readability factor

Readable extracts are hard to be achieved. A readable document is one in which sentences are highly related to their proceeding sentences. first sentence and second sentence are related to each other with a high similarity, same for second and third sentences, and so on. In fact, a readable summary, as we define, is made up of sentences which form a *smooth* chain of sentences.

Suppose the readability of summary s with length S, say R_s, is

$$R_s = \sum_{0 \le i < S} W(s_i, s_{i+1})$$

Therefore, the readability factor of summary s, is computed like this:

$$\mathrm{RF}_s = \frac{R_s}{\max_{\forall i} R_i}$$

It should be noted that we have made the assumption that the summary length is fixed and the maximum is computed among all possible summaries of that summary length.

Finding this maximum can be done in polynomial time. Suppose the summary length is S, so the goal of finding the most readable summary is equal to finding a path of length S with maximum weight in the document graph.[2]

3.3.1 Longest path

It is time to find the longest path with a definite number of nodes from the DAG, which is created above. Here, we intend to find the maximum path with a fixed length.

It is simple if we consider the summary to be $s_{s_1}, s_{s_2}, \ldots, s_{s_S}$, The goal is to maximise

$$\mathrm{sim}_{s_{s_1}, s_{s_2}} + \mathrm{sim}_{s_{s_2}, s_{s_3}} + \cdots + \mathrm{sim}_{s_{s_{S-1}}, s_{s_S}}$$

Solving the problem is simple.

A 2-D array is used, say A, where ith row shows s_i and jth column shows path of length j. So, $A_{i,j}$ = Weight of the Longest Path to the ith sentence with the length j, this means:

$$A_{l,j} = \max_{i<l} \left(A_{i,j-1} + sim_{i,l} \right)$$

Figure 2 shows the matrix.

Here is the algorithm to fill the array. Algorithm 1 takes s, the length of the desired summary, and the similarity matrix of graph representation as input and fills up the cells of a matrix like Figure 2. At last, it finds the weight of the path of length s with the highest weight.

To find the smoothest extract with k sentences, we must find the path with maximum weight of the length k, that is the cell with maximum value in the kth column of A. To keep track of path construction, simply another Matrix is needed. So, we can have the path in polynomial time easily.

Figure 2 Dynamic solution, A_{ij} is the cost of heaviest path ending at node i with j edges

	0	**1**	**2**	**3**	...
0	0	0	0	0	0
1	0	0	0	0	0
2	0	$Sim_{1,2}$	0	0	
3	0	$Max(Sim_{1,2}, Sim_{1,3})$	$Sim_{1,2} + Sim_{2,3}$		
4	0	.			
5	0	.			
. . .		.			

Algorithm 1 Finds path with the highest weight

Require: N : *length of document*
Require: s : *required length of summary*
Require: Sim : *input Similarity matrix of graph*
Ensure: Path with length s and highest weight

1: $m \leftarrow N$
2: $n \leftarrow N$
3: **for** $i = 0$ to m **do**
4: **for** $j = 0$ to n **do**
5: $A[i, j] \leftarrow 0$
6: **end for**
7: **end for**
8: **for** $j = 1$ to n **do**
9: **for** $i = 1$ to m **do**
10: **for** $k = 1$ to $i - 1$ **do**
11: **if** $A[i, j] < A[k, j - 1] + \text{Sim}(k, i)$ **then**
12: $A[i, j] \leftarrow A[i, j - 1] + \text{Sim}(k, i)$
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: Return A

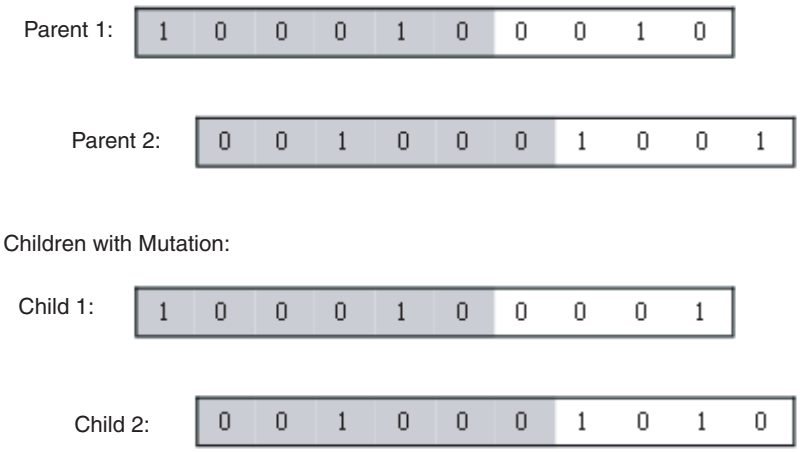
4 Genetic algorithm

The use of GA, was an idea to avoid problems with local search techniques. Local search may find a local maximum and declare it as the answer.

In our problem, the goal is to find a summary with high readability, high cohesion and high topic relation.

One approach to find such a summary is using GA. To do so consider a vector where ith element in that vector is 1 if s_i is in summary, and is 0 if s_i is not included in the summary. Suppose, also the number of sentences in summary is desired to be S, and $S < N$, where N is total number of sentences. Construct some vectors randomly with S, $1s$ and $N - S$, $0s$. Use this population to make children by crossover and a single mutation. One child should be generated based on patterns of both parents. It should inherit half pattern of one parent and other half from the other parent as shown in Figure 3.

Figure 3 Genetic Algorithm on summary population



After that, all parents and children are put together and a fitness function is used to choose half fittest of the new population. These half are next round's parents. The procedure continues until no more fitter summary is generated. Here, we show how to construct the fitness function.

4.1 Fitness function

The fitness function is a flexible one, that means, it has parameters to be adjusted by the user need. One may desire a summary with the highest readability while another may like to have a summary in which sentences are highly related to the topic, and readability is not a matter at all.

To have such a fitness function, we design a function which is the weighted average of three factors:

$$F = \frac{\alpha \times \mathrm{TRF} + \beta \times \mathrm{CF} + \gamma \times \mathrm{RF}}{\alpha + \beta + \gamma}$$

α, β, γ are real numbers and are defined by user. Some notes about defining these coefficients is explained in the following commentary section.

TRF, CF , RF are all between 0 and 1, so this composition of them results in a real number between 0 and 1.

4.1.1 Commentary

In this part, we focus on users. Defining the coefficients, α, β, γ may be a little tricky, and users may become a little baffled in defining them. In fact, it depends on the type of the user and also on the needs of the user. With the guidelines below, it seems easy to decide about the coefficients.

It is clear that with larger values of α in comparison with the other coefficients, the summary will be of much *topic relation*, than readability or cohesion. On the other hand, with larger magnitudes of β and γ, the summary will be more *cohesive* and *readable*, respectively.

Sometimes the document does not have a title, so α must be set to zero. In such cases, the method might be used to extract a title for the text.

If the user tends to extract a summary, which will be later used as a separate document, then cohesion should be of high value and thus, β should be higher in comparison with other coefficients. Besides, if smoothness and readability is the main concern, the user is encouraged to set γ a much higher value than β and α.

In fact, the user can play with these coefficients and test various values for them to find the fittest values for them, unless of course no corpus is available, which in that case, a learning method could be useful to find suitable values for them.

4.2 Comparison with greedy approaches

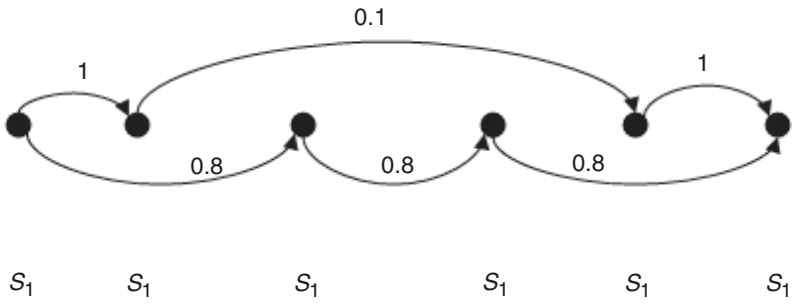
Sometimes, greedy approaches are helpful, but on some occasions, they do not always find best answers. For example, a greedy algorithm-based solution may find a wrong answer as shown below. A document of six sentences is presented, and a summary of four sentences is desired. Some edges are not displayed for simplicity. Figure 4 shows an example. Suppose that the similarity matrix is as follows:

$$\mathbf{Sim} = \begin{pmatrix} 0 & 1 & 0.8 & 0.1 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0.8 & 0 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0.8 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The greedy search finds $\mathrm{Sim}_{1,2}$, $\mathrm{Sim}_{5,6}$ the most weighted edges. These two selections make 4 vertices and result the sequence: s_1, s_2, s_5, s_6 for which the sum of similarities is: $\mathrm{sim}_{1,2} + \mathrm{sim}_{1,5} + \mathrm{sim}_{1,6} + \mathrm{sim}_{2,5} + \mathrm{sim}_{2,6} + \mathrm{sim}_{5,6} = 1 + 0.1 + 0.1 + 0.1 + 0.1 + 1 = 2.4$, and sum of respective similarities, $\mathrm{sim}_{1,2} + \mathrm{sim}_{2,5} + \mathrm{sim}_{5,6} = 1 + 0.1 + 1 = 2.1$ But the sequence s_1, s_3, s_4, s_6 with sum of similarities, $\mathrm{sim}_{1,3} + \mathrm{sim}_{1,4} + \mathrm{sim}_{1,6} + \mathrm{sim}_{3,4} + \mathrm{sim}_{3,6} + \mathrm{sim}_{4,6} = 0.8 + 0.1 + 0.1 + 0.8 + 0.1 + 0.8 = 2.7$ and sum of respective similarities, $\mathrm{sim}_{1,3} + \mathrm{sim}_{3,4} + \mathrm{sim}_{4,6} = 0.8 + 0.8 + 0.8 = 2.4$ is a better choice for a summary, either regarding readability or cohesion factors.

GA, while adopted with a good mutation tends to create results close to the optimum. The result could be a highly readable one if γ is set to be close to 1.

Figure 4 Greedy search inappropriate results



5 Evaluation

We have used an intrinsic method to evaluate our summarisation algorithm (Edmunson, 1969; Kupiec et al., 1995; Marcu, 1997). The typical approach is to create 'ideal' summaries, either written by professional abstractors or by merging summaries provided by multiple human subjects (Barzilay and Elhadad, 1997). Once automatic generated summaries are in hand, they are compared to 'ideal' summaries. Precision and Recall are computed and the quality of summaries is measured.

We used a similar approach to evaluate summaries generated by our GA.

10 sets of documents, each containing average 7 separate documents were randomly selected from 'Document Understanding Conference' (DUC), 2002 data. For each set, two summaries were created. One of them with approximately 400 words, and the other with 200 words.

The 400 word summary was compared to two human-made 400 word summaries, and the 200 word summary was compared to two human-made 200 word summaries given by DUC.

5.1 Similarity of model summaries

The similarity of model summaries[3] was given by DUC. Results show how human-made summaries differ from each other (see Table 1).

For each set, two model summaries of a specific length (200 or 400 words) were compared. Precision and recall are computed, considering one of them as model and the other to evaluate. It is obvious that if the roles are changed, the values for precision and recall would also change.

The results show that two model summaries for each set are created from two different points of view. This is known from the analysis on the data on Table 1 and the fact that two model summaries are not tightly similar.

5.2 Evaluation algorithm

Because the way, in which human-made summaries were made, was unknown to us to discover what α, β and γ might be, we tested these coefficients in a discrete range. That means for the values:

$$0 \leq \alpha \leq 1$$

$$0 \leq \beta \leq 1 - \alpha$$

$$\gamma = 1 - \alpha - \beta$$

That are calculated according to algorithm 2.

Table 1 Human-made summaries similarity

Set No.	*Size (W)*	*Precision*	*Recall*
D061j	200	0.300	0.300
	400	0.368	0.368
D062j	200	0.250	0.250
	400	0.333	0.267
D063j	200	0.111	0.143
	400	0.211	0.286
D067f	200	0.143	0.143
	400	0.294	0.294
D071f	200	0.455	0.556
	400	0.619	0.650
D074b	200	0.000	0.000
	400	0.067	0.071
D097e	200	0.375	0.429
	400	0.214	0.200
D113h	200	0.500	0.444
	400	0.444	0.375
D070f	200	0.222	0.222
	400	0.471	0.500
D066j	200	0.333	0.333
	400	0.150	0.167

In each iteration, for the corresponding α, β, γ the fitness function is made up, and using that fitness function, a summary is created, precision and recall of which are available in Table 2. The average and max values of precision and recall for a summary in Table 2, are computed from the results in the iterations of the above algorithm.

Again, we should emphasise that this algorithm is just for our evaluation of our method, because we had no idea what α, β, γ for model summaries might be. So, we tested 10 values of each, to calculate an average and maximum, to see how efficient our method is. An ordinary user, with some intentions for summarising a document, does not need this algorithm, because he can easily set α, β, γ to his desired values to have a summary with features of his will.

5.3 Genetic algorithm performance

We used past DUC results to compare our results with those of previous researchers who participated in DUC 2002.

Algorithm 2 Calculate max and average precision and recall for summarising a single document

Require: s, $input\ summary$
Ensure: $MaxPrecision$, $MaxRecall$
Ensure: $AveragePrecision$, $AverageRecall$
1: $TRF \leftarrow TRF_s$
2: $CF \leftarrow CF_s$
3: $RF \leftarrow RF_s$
4: $Max\_P \leftarrow 0$ {Max Precision}
5: $Max\_R \leftarrow 0$ {Max Recall}
6: $Average\_P \leftarrow 0$ {Average Precision}
7: $Average\_R \leftarrow 0$ {Average Recall}
8: **for** $i = 0$ to 10 **do**
9: **for** $j = 0$ to $10 - i$ **do**
10: $\alpha \leftarrow i/10$
11: $\beta \leftarrow j/10$
12: $\gamma \leftarrow 1 - \alpha - \beta$
13: $F \leftarrow (\alpha \times TRF + \beta \times CF + \gamma \times RF)/(\alpha + \beta + \gamma)$
14: Run GA and Compute Precision and Recall
15: **if** $precision > Max\_P$ **then**
16: $Max\_P \leftarrow precision$
17: **end if**
18: **if** $Recall > Max\_R$ **then**
19: $Max\_R \leftarrow Recall$
20: **end if**
21: $Average\_P \leftarrow Average\_P + precision$
22: $Average\_R \leftarrow Average\_R + Recall$
23: **end for**
24: **end for**
25: $Average\_P \leftarrow Average\_P/66$
26: $Average\_R \leftarrow Average\_R/66$

As the results are shown in Table 2, in most cases, the best precision and recall obtained by our GA, are better than the DUC 2002, competitors' results. In almost all cases, the average precision and recall of several runs in GA, leads to better results than the average precision and recall of DUC 2002 challengers.

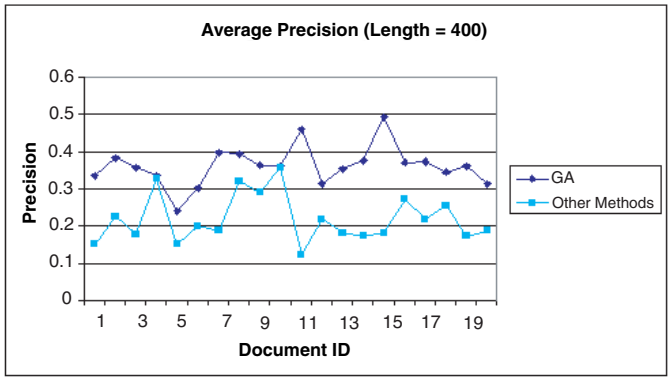
Let us briefly describe the fields in Table 2. First column is the ID of 'Document Set'. Second column shows the code for the 'Model Summariser', which is an 'ideal' summariser. There are exactly two extract model summariser for each set of documents. Third column shows the length of the summary of a model summariser, which at the same time is our bound to make summaries. For example, if the length is 200 words, it means that the model summary is 200 words in length, and so is our generated summary, which is being evaluated with that model. The next couple of columns show Average precision and recall for GA, and Past results of DUC 2002, respectively. Last two couples of columns are similar to previous ones but they show the maximum precision and Recall reached by GA and the methods on DUC 2002 as shown in Table 2.

Table 2 Evaluation results

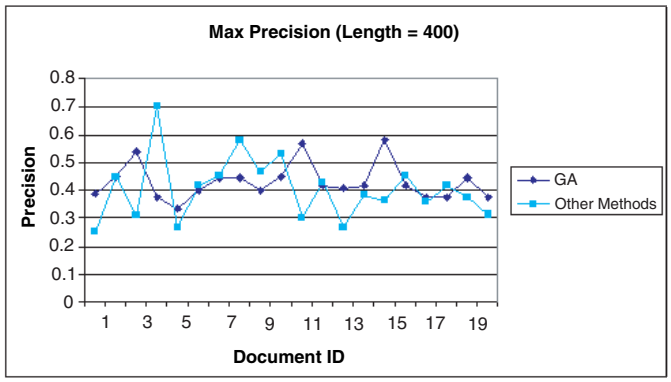
Set No.	*Model*	*Size(W)*	*GA*	*GA*	*Other*	*Other*	*GA*	*GA*	*Other*	*Other*
			Average Precision	Average Recall	Average Precision	Average Recall	Max Precision	Max Recall	Max Precision	Max Recall
D061j	B	400	0.337	0.263	0.152	0.126	0.389	0.3	0.25	0.263
		200	0.342	0.323	0.106	0.08	0.36	0.4	0.222	0.2
	I	400	0.382	0.3	0.227	0.184	0.45	0.35	0.45	0.474
		200	0.344	0.32	0.225	0.17	0.46	0.4	0.429	0.3
D062j	A	400	0.358	0.346	0.179	0.2	0.54	0.5	0.308	0.333
		200	0.408	0.362	0.076	0.062	0.6	0.4	0.167	0.125
	G	400	0.336	0.312	0.329	0.286	0.375	0.4	0.7	0.467
		200	0.358	0.266	0.188	0.162	0.4	0.4	0.429	0.375
D063j	C	400	0.242	0.219	0.15	0.115	0.334	0.264	0.267	0.211
		200	0.286	0.223	0.1	0.088	0.286	0.223	0.3	0.333
	E	400	0.302	0.367	0.202	0.199	0.4	0.429	0.417	0.357
		200	0.286	0.286	0.091	0.085	0.286	0.286	0.25	0.143
D067f	A	400	0.398	0.294	0.188	0.129	0.445	0.357	0.455	0.294
		200	0.347	0.375	0.167	0.128	0.428	0.5	0.4	0.286
	I	400	0.396	0.295	0.321	0.229	0.445	0.4	0.583	0.412
		200	0.354	0.428	0.249	0.2	0.428	0.429	0.5	0.429
D071f	A	400	0.364	0.346	0.292	0.218	0.4	0.381	0.467	0.476
		200	0.33	0.227	0.266	0.218	0.375	0.273	0.455	0.455
	B	400	0.363	0.363	0.36	0.275	0.45	0.45	0.533	0.45
		200	0.366	0.305	0.209	0.188	0.429	0.334	0.4	0.444
D074b	A	400	0.461	0.263	0.123	0.106	0.57	0.334	0.3	0.2
		200	0.5	0.26	0.058	0.04	0.625	0.3	0.167	0.1
	E	400	0.315	0.246	0.22	0.192	0.416	0.291	0.429	0.429
		200	0.338	0.263	0.172	0.171	0.429	0.289	0.5	0.429
D097e	A	400	0.355	0.342	0.181	0.178	0.41	0.41	0.267	0.268
		200	0.347	0.375	0.07	0.062	0.428	0.5	0.25	0.25
	J	400	0.378	0.344	0.173	0.16	0.416	0.4	0.385	0.333
		200	0.354	0.428	0.122	0.114	0.428	0.429	0.333	0.286
D113h	A	400	0.496	0.378	0.184	0.16	0.583	0.428	0.364	0.333
		200	0.459	0.428	0.079	0.062	0.571	0.571	0.286	0.25
	I	400	0.372	0.363	0.273	0.219	0.416	0.385	0.455	0.438
		200	0.381	0.417	0.122	0.077	0.428	0.5	0.286	0.222
D070f	G	400	0.375	0.352	0.22	0.188	0.375	0.353	0.357	0.294
		200	0.3	0.33	0.168	0.122	0.3	0.33	0.5	0.33
	J	400	0.345	0.345	0.256	0.231	0.375	0.375	0.417	0.375
		200	0.3	0.33	0.186	0.144	0.3	0.33	0.375	0.33
D066j	C	400	0.361	0.34	0.173	0.125	0.445	0.45	0.375	0.3
		200	0.331	0.348	0.103	0.077	0.4	0.5	0.375	0.33
	I	400	0.315	0.294	0.189	0.144	0.375	0.35	0.312	0.278
		200	0.275	0.324	0.179	0.133	0.3	0.429	0.5	0.444

Figure 5 briefly describes the results. X axis in each chart is the ID of 'Document Set'. There are exactly two extract model summariser for each set of documents, and so for 10 sets, we have 20 results.

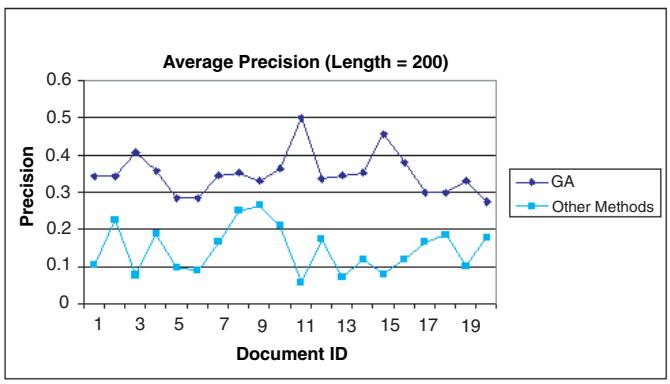
Figure 5 Run Results (a)average Precision of summaries of length 400, (b) maximum Precision of summaries of length 400, (c) average Precision of summaries of length 200 and (d) maximum Precision of summaries of length 200 (see online version for colours)



(a)

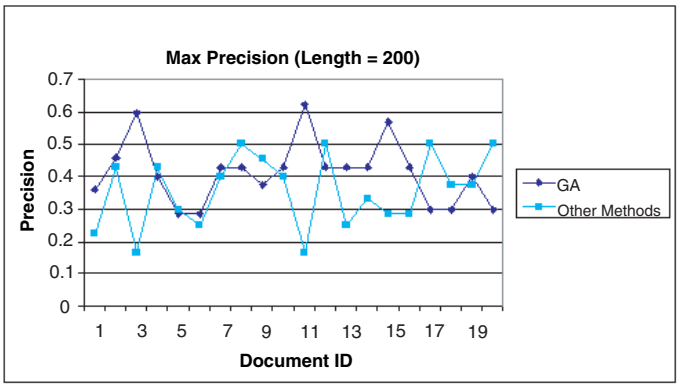


(b)



(c)

Figure 5 Run Results (a)average Precision of summaries of length 400, (b) maximum Precision of summaries of length 400, (c) average Precision of summaries of length 200 and (d) maximum Precision of summaries of length 200 (see online version for colours) (continued)



(d)

5.4 *Implementation issues*

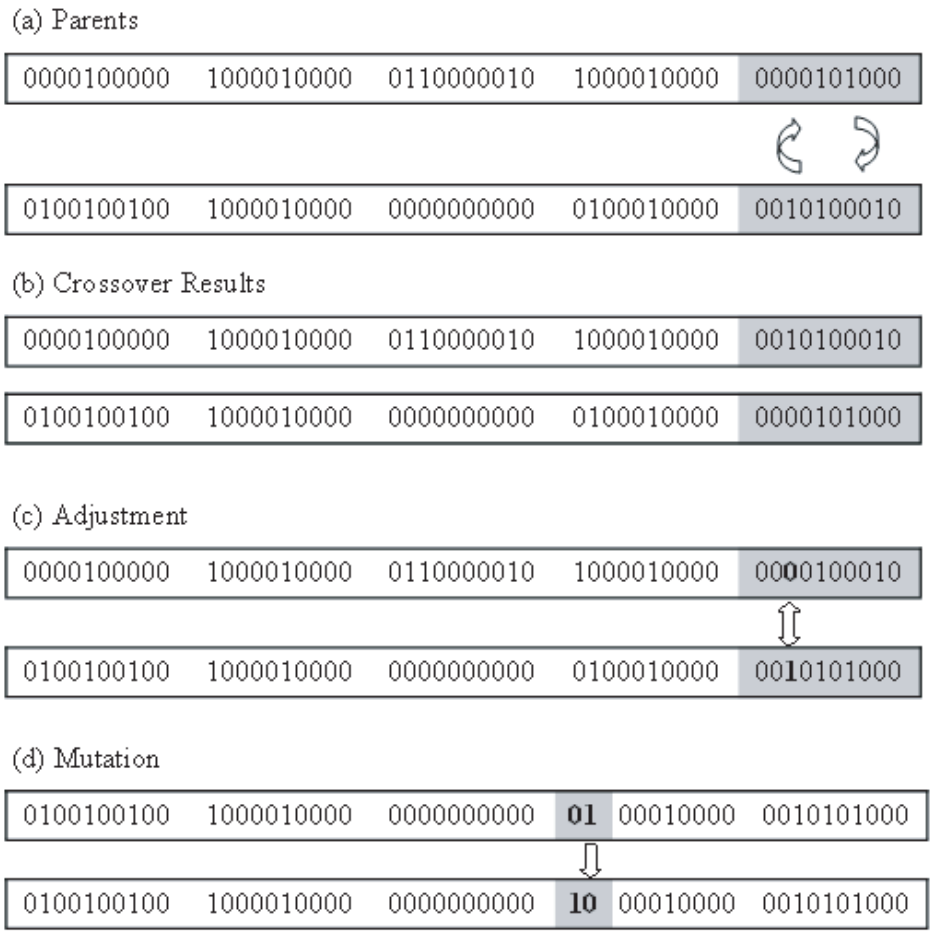
Implementing GA for such a coding, needs a lot of attention, Search domain is not a 'soft' one in this problem. By this expression we mean that, a little change in a summary vector may cause a totally different summary. Much often, it is very far from the goal state.[4] So, it is not a good idea to have a crossover that changes half of a parent with that of another. This kind of crossover on two 'good' parents may create two children which are very poor in terms of fitness function. So, big changes are not good for this problem, and may change the GA to a simple random change.

For each set of α, β, γ we ran the GA with a fixed point crossover that swaps the last ith part of two parents, where i is randomly chosen between 5 and 10. A sample is shown in Figure 6(a) where $i = 5$. The two parts are simply changed with each other. In this transition, it is taken care not to exceed summary length limit in terms of sentences. Of course, if it does so, and number of 1s in a child is more than the limit[5], some extra 1s are cleared in a way that the child closes to its parent. We do not change any arbitrary 1 to zero. The method is to undo the swap of a 1 with other parent's corresponding bit as shown in Figure 6(c). This is done until the number of 1s in both children is equal, and the two created children have exactly the same number of 1s, equal to their parents.

We also did a *single* mutation. An arbitrary '1' bit is chosen and is swaped with one of its neighbour '0' bit. If such a neighbour dos not exist, another '1' bit will be chosen as illustrated in Figure 6(d). This reduces the probability of big changes in summary fitness, because sentences which are next to each other in chronological order of the text, may highly talk about the same idea.

Using the crossover and mutation functions, as described above, for each two parents, two offsprings are created. This will lead to a population size twice as much as the older one. The new population is sorted due to the fitness function, and the best half are chosen as parents for the next iteration of the GA. This process will continue until the fitness value of individuals in population converge.

Figure 6 Implementation issues sample. Number of sentences is 50 and 10 of them are chosen for summary: (a) shows two parents, (b) Crossover has been done on two parents of part a, but number of 1s, is not 10, (c) Adjustment on two children to correct number of 1s and (d) Mutation



6 Conclusion and future work

This research shows the Sentence Selection-Based Genetic Algorithm method. The results are better in many cases compared to other methods, and in those which are not at the best standings, they are good enough to call this method efficient. The best performance of the system would be when a user exactly knows how to define α, β and γ, so that the result and summary would suit best for his/her requirements. Work is being done to make a learning method for these coefficients that would learn how to extract documents in a system without user interaction. Also, some other factors that make a summary a good one, are not involved in this modelling. Some of them are features that considering them, will improve the quality of the resultant summary. 'Sentence length' (Kupiec et al., 1995), 'position of sentences' (Mani and Bloedorn, 1998; Teufel and Moens, 1999; Kupiec et al., 1995), 'Proper names' they contain(Kupiec et al., 1995), etc. are some of them. If these features are involved in mutual similarity of sentences, the results would be much better.

One of the applications of this method is the processing of online news and creating summary for time saving of human users. Since there is a vast amount of news which is available on the internet, one may use such an algorithm to make news summaries to be consumed by human users, computer agents or make a compact archive of them.

References

Baeza-Yates, R. and Ribeiro-Neto, B. (1999) *Modern Information Retrieval*, Addison Wesley.

Barzilay, R. and Elhadad, M. (1997) 'Using lexical chains for text summarization', *Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97)*, ACL.

Brandow, R., Mitze, K. and Rau, L.F. (1995) 'Automatic condensation of electronic publications by sentence selection', *Information Processing and Management*, Vol. 31, No. 5, pp.675–685.

Edmunson, H. (1969) 'New methods in automatic extracting', *Journal of ACM*, Vol. 16, No. 2, pp.264–285.

Goldstein, J., Kantrowitz, M., Mittal, V.O. and Carbonell, J.G. (1999) 'Summarizing text documents: sentence selection and evaluation metrics', *Research and Development in Information Retrieval*, pp.121–128.

Hovy, E. and Lin, C. (1997) 'Automated text summarization in summarist', *ACL/EACL-97 Workshop on Intelligent Scalable Text Summarization*, pp.18–24.

Kupiec, J., Pedersen, J.O. and Chen, F. (1995) 'A trainable document summarizer', *Proceedings of 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.68–73.

Luhn, H. (1958) 'The automatic creation of literature abstracts', *IBM Journal of Research and Development*, Vol. 2, No. 2.

Mani, I. and Bloedorn, E. (1998) 'Machine learning of generic and user-focused summarization', *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI/IAAI*, pp.821–826, AAAI Press/The MIT Press.

Marcu, D. (1997) 'From discourse structures to text summaries', *The Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pp.82–88, Madrid, Spain.

Mitra, M., Singhal, A. and Buckley, C. (1997) 'Automatic text summarization by paragraph extraction', *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pp.39–46.

Salton, G. and Buckley, C. (1998) 'Term-weighting approaches in automatic text retrieva', *Information Processing and Management*, Vol. 24, No. 5, pp.513–523.

Salton, G., Singhal, A., Buckley, C. and Mitra, M. (1996) 'Automatic text decomposition using text segments and text themes', *UK Conference on Hypertext*, pp.53–65.

Silla, J., Nascimento, C., Pappa, G.L., Freitas, A.A. and Kaestner, C.A.A. (2004) 'Automatic text summarization with genetic algorithm-based attribute selection', *Lecture Notes in Artificial Intelligence*.

Teufel, S. and Moens, M. (1999) 'Argumentative classification of extracted sentences as a first step towards flexible abstracting', in I. Mani and M.T. Maybury (Eds). *Advances in Automatic Text Summarization*, pp.155–171, The MIT Press.

Notes

[1]Cohesion Matrix.

[2]Weight of a path is the sum of weights of its edges.

[3]Human made summaries.

[4]Consider any summary vector as a state, the fittest summary is the goal and GA tries to reach that goal.

[5]In that case, the other child will have less 1s because the total number of 1s is constant.