Biological Model For Information Retrieval

Moheb R. Girgis[1], Abd El-Mgeid A. Aly[2], Bahgat A. Abdel Latef, Boutros R. El-Gamil[3]
Department of Computer Science, Faculty of Science
Minia University, El-Minia, Egypt.
[1]mrgirgis@mailer.eun.eg
[2]abdelmgeid@yahoo.com
[3]boutrosgameil@yahoo.com

ABSTRACT: *Heuristic methods based on biological aspects have been successfully used in many computer science areas of research, including information retrieval (IR). This let us ask: why there is no biological environment for the problem of information retrieval. In this paper, we tried to introduce a new biological environment and model for information retrieval that broad the modeling concept from the mathematical formula that simulates the basic elements of IR problem to their dual schemas in biology. It's a new way of thinking of, and dealing with, the problem of information retrieval.*

Categories and Subject Descriptors
H.3.3[Information Search and Retrieval]; I .6 [Modeling and Simulation] Multimedia Databases: **J.3 [Life and Medical Sciences]**

General Terms
Information Retrieval, Biological modeling, Genetics application, Mathematical model, Simulation

Keywords: Information retrieval, Biological environment, Biological model, Genetic algorithms

Received 30 November 2006; Revised 15 February 2007; Accepted 12 March 2007

1. Introduction

Probabilistic methods are relatively recent in computer science, but their range of applications has increased rapidly in many research areas, including information retrieval. Genetic algorithms (GAs) specifically have been used by many researches to solve IR problems. In this vein, the main directions concern modifying the document indexing (Gordon [6]; Blair [2]), the clustering problem (Raghavan and Agarval, [11]; Gordon [7]) and improving query formulation (Yang et al. [18]; Perty et al. [10]; Chen [3]; Horng and Yeh [8]). Also, there are good experiments to improve GA operators that are applied to IR (Vrajitoru [16]; Vrajitoru [17]).

However, every time we adopt a heuristic method to an IR problem, we force the given method in order to suit the task at hand. This paper introduces a new ideology of dealing with the problem of information retrieval. It builds a new IR environment based on biological aspects. This model used the similarity between textual material and the biological chromosome. From this point, we move, step by step, through document representation, term indexing, term classification, and search strategy. Finally, we tested two retrieval models on our new IR biological schema.

The paper is organized as follows: Section 2 shows the similarity between the textual document and the biological chromosome. Section 3 describes biological environment phases we adapt in our schema. Section 4 describes query manipulation and section 5 describes the search strategy of our system. Our biological model for IR is introduced in section 6. Experiments are given in section 7. In section 8, we compared our model with two probabilistic IR models, which are vector space model (VSM) with new weightings, and Okapi model. In section 9, we introduced some modifications to our model to improve its performance.

2. Similarity between Biological Chromosome and Textual Document

By comparing the structure and functionality of the biological chromosome and the textual document, we can discover great similarity between them. As the chromosome consists of a series of nucleotides, the document consists of a series

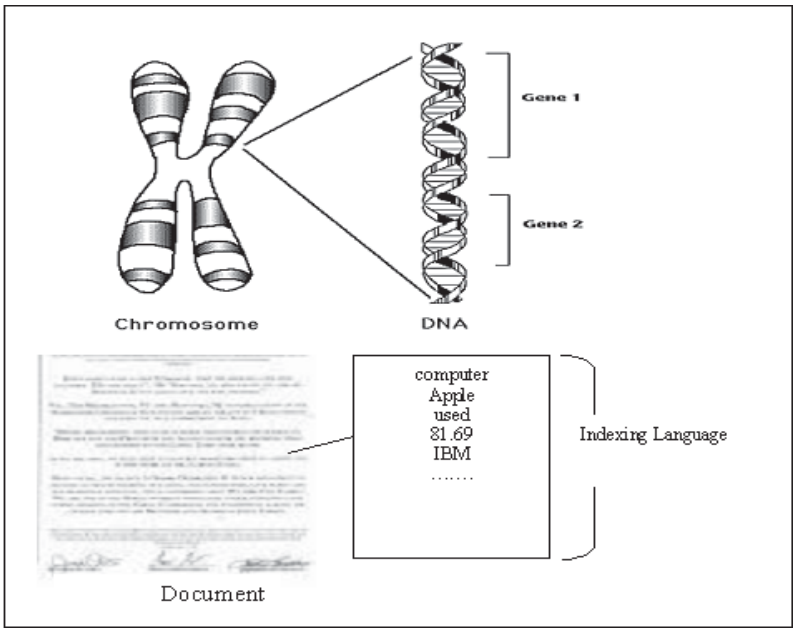

Figure 1. Similarity between biological chromosome and textual document

of tokens (words). Also, within the chromosome, a sub seriesof nucleotides, with some known function, called a gene, corresponds to the group of tokens that we extract from the document and represent it within the IR system, called the indexing language. Figure 1 shows the similarity between both components.

If we give a more deeply looking to the structure of both chromosome and document, we find that as the chromosome consists of four types of nucleotides (thymine, cytosine, adenine, guanine), the document consists of four basic types of tokens (lower-case, capitals, abbreviations, and digits). Figure 2 shows this duality between both chromosome and document.

3. Biological Environment Phases

3.1 Text Analysis and Gene Extraction

In order to extract the indexing language (gene) of the document, we follow the steps given below [5]:

3.1.1 Eliminating special characters and stop words

Firstly, we need to eliminate special characters, (&, /, $, ...), from the document. Also, we eliminate stop words. That is, the high frequency function words that are insufficiently specific to represent document content (i.e. are, each, ...)

3.1.2 Isolating digits, capitals, and abbreviations

After that, the remaining tokens of document can be classified linguistically into four categories. These categories are digits, capitals, abbreviations, and lower-case words. Our model depends basically on these types of tokens. So, we isolate digits, capitals, and abbreviations from the context, and store them as part of the document representation.

3.1.3 Lower-case tokens

The only type of tokens left after the above 2 steps is the lower-case words, which represents the core of most textual materials. For this category, a suffix-stripping routine (stemmer) is applied to reduce significant words in the collection of words to word stems. Instead of throwing suffixes away, we save the suffix of each stem to be used later in term representation.

3.1.4 Calculating frequency

After stemming lower-case tokens, we calculate the frequency of each stem. That is, number of occurrences of each stem within the document. And for each stem, we store its different suffixes generated from the stemming process.

3.1.5 Assistant Factor

The assistant factor (AF) of a given token is defined to be the most frequent suffix appended to the given stem all over the document. This parameter is obtained by calculating the frequency of each suffix of the stem. If one suffix frequency is higher than the others, it is declared as the AF of the given stem. Otherwise, if more than one suffix have the same highest frequency, then the AF parameter is set to zero (Null). After assigning assistant factor parameter to each stem (nucleotide), we can construct the first series of nucleotides as shown in Figure 3. Each nucleotide (nuc) is consisted of two parts. The stem part and the AF part, which represent the significant suffix of the stem.

3.1.6 Selecting gene nucleotides

Inspired from biological facts, each chromosome contains hundreds, or may be thousands, of nucleotides. However, only few number of them can construct a series with well known, and useful function (gene). As we want our gene to be as concentrated as possible, i.e. consists only of the important (most frequent) nucleotides. We arrange nucleotides decreasingly according to nucleotide frequency. According to the size of our collections, we specify a maximum number of nucleotides, (limit), to represent the document gene over the actual number of nucleotides (nucnum) found so far. For this purpose, we adapted rule that is based on predefined number of nucleotides to be applied:

```
If (nucnum< 150) Then
limit= 10;
Else If (nucnum>= 150 AND nucnum< 250) Then
        limit= 15;
Else If (nucnum>= 250 AND nucnum< 400) Then
        limit= 20;
Else
limit= 25;
```

The system begins to take nucleotides of higher frequency while the given limit is not reached. If the system comes to a level of frequency that its number of nucleotides exceeds the given limit, the system selects the required number of nucleotides randomly from that level. Figure 2 shows a block diagram of this phase.

Note that this number of nucleotides acts as part of document representation in our environment, beside other parts (capitals, abbreviations, ...).

In this paper, Head section represents 70% of nucleotides number, Body section is 20%, and Tail section is 10%.

Note that, our genetic classification depends on frequency variation between nucleotides. But, if all nucleotides frequencies are equal, all nucleotides will be classified as Head ones.

3.3 Genetic Map

Over a set of documents, where each one is represented by a gene, we order nucleotides of all documents alphabetically. After that, we calculate the document frequency, (docfreq), of each nucleotide, i.e., the number of documents in which each nucleotide occurs. Then, we construct a corpus of genes (called the genetic map), that contains all genetic information about nucleotides in each gene. Figure 5 shows how this map looks like.

In this figure, nucleotide axis represents nucleotides of documents collection ordered alphabetically. For each nucleotide, there are three pieces of information connected to it: Gene_ID (the document number/index in the collection), AF (the assistant factor of nucleotide in that document), and Position (the section on which the nucleotide appears in that document).

From programming point of view, we can represent the genetic map by two data files: nucleotides file and statistics file. Nucleotides file holds nucleotides of the map. Each nucleotide data structure has 3 fields: name (which holds nucleotide stem), docfreq (which holds nucleotide document frequency, and address (which holds the reference of statistical part of nucleotide into statistics file). Figure 6 shows the implementation of the genetic map.

3.4 Other Maps

Beside lower-case tokens, there are other linguistic components (digits, capitals, and abbreviations) and/or

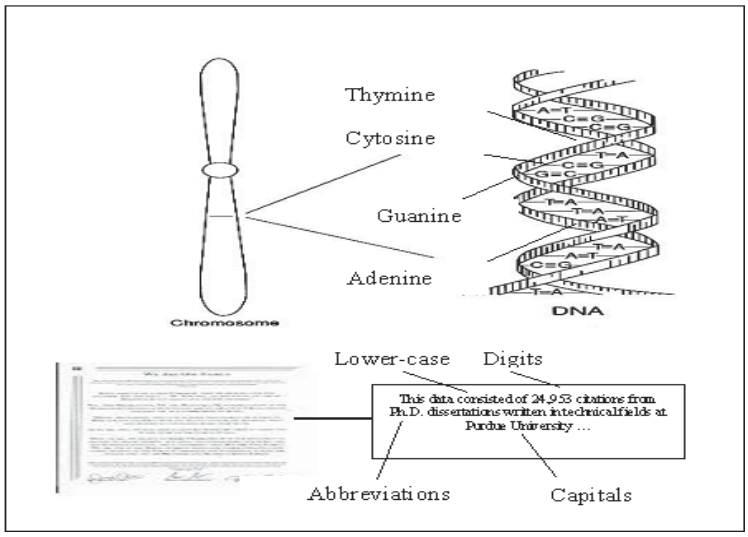

Figure 2. Structural similarity between biological chromosome and textual document

statistical components (Author, Publisher … etc.), that could be included in or attached to a given document. So, we construct three linguistic maps (Capitals map, Abbreviations map. and Digital map) as well as one statistical map (Authors map) in addition to our basic genetic map. Figure 6 shows the general structure of a non-genetic map

4. Query Manipulation

A query is manipulated just the same as a document. So, for a query we obtain query nucleotides, query capitals, query abbreviations, and query digits.

5. Search Strategy

Our search strategy is shown in Figure 8. Firstly, we search for abbreviations and digits in abbreviations and digital maps, respectively. Then, we search for capital nucleotides in authors map and capitals map. After that, we perform a transformation process, by stemming capital nucleotides, turning them into lower-case nucleotides, and search for them as B (Body) nucleotides within the genetic map. Finally, we search for query nucleotides in the genetic map.

6. Biological Model for Information Retrieval

In this section, we introduce a new formula for information retrieval.

6.1 Maximum likelihood estimate

A maximum likelihood estimate can be defined as in (Song and Croft [14]):

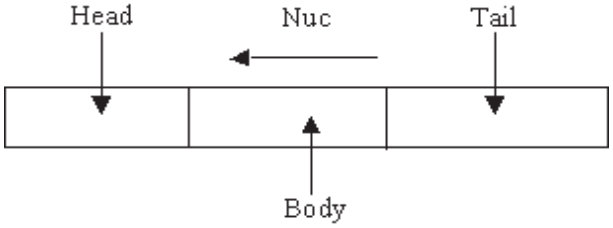

Figure 4. Gene diagram after genetic classification

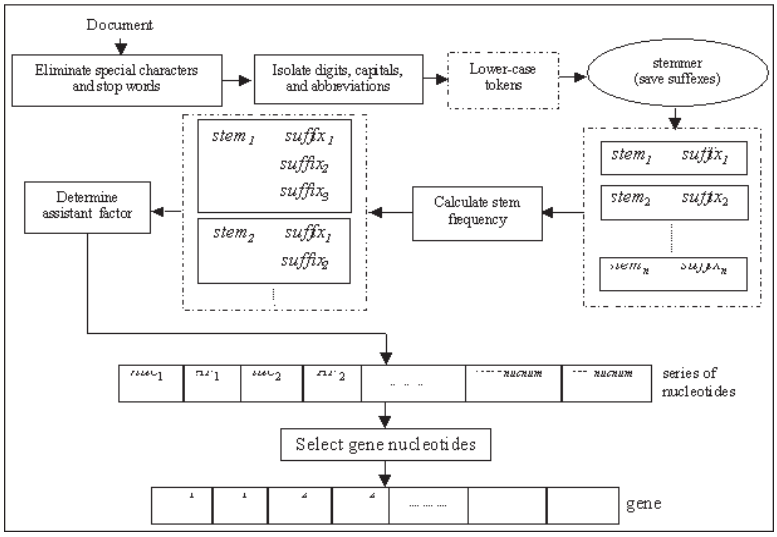

Figure 3. A block diagram of phase 1 of the biological model

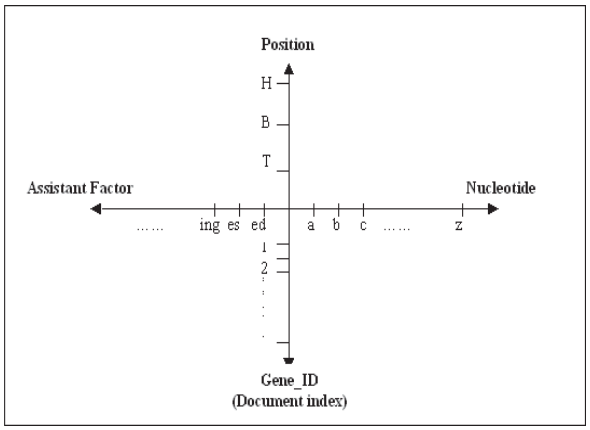

Figure 5. Genetic map

$$P_{mle}(T_j, D_i) = \frac{tf_{ij}}{N_{D_i}} \quad (1)$$

where tf_{ij}, is the number of occurrences of term $T_j = 1,2,...,t$, in document D_i, and N_{D_i} is the total number of term occurrences in a document D_i, (or in a query).

This formula can be generalized as:

$$P_{mle}(T_j, D_i) = \frac{Tf_{ij}}{gene\_length_{D_i}} \quad (2)$$

where Tf_{ij}, is the number of different tokens related to a given token type T (nucleotides, capitals, digits,... etc) in document D_i, and $gene\_length_{D_i}$ is the total number of tokens of all types in document D_i.

By multiplying equations (1) and (2), we obtain:

$$P_{general}(T_j, D_i) = \frac{tf_{ij}}{N_{D_i}} \times \frac{Tf_{ij}}{gene\_length_{D_i}} \quad (3)$$

6.2 Juxtaposition vs. discrimination

Since our model is based on token categorization, and we search a variety of maps, we need to juxtapose equation (3) to suite each token type and frequency. If we assume that 'five' is the average number of maps that can be supported by this model, an exponent of 1/5 (0.2) is expected to be fair:

$$P_{general}(T_j, D_i) = \left(\frac{tf_{ij}.Tf_{ij}}{N_{D_i}.gene\_length_{D_i}} \right)^{0.2} \quad (4)$$

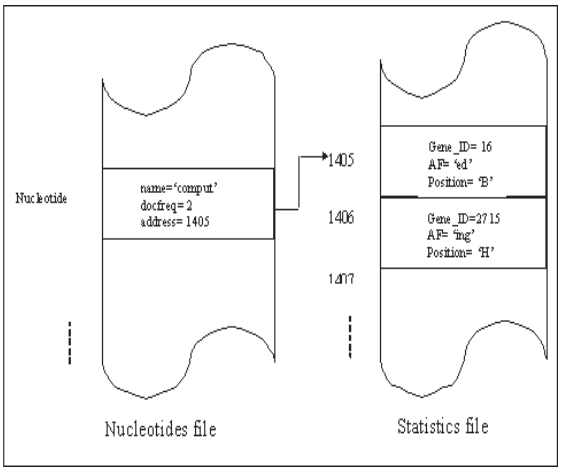

Figure 6. Implementation of the genetic map

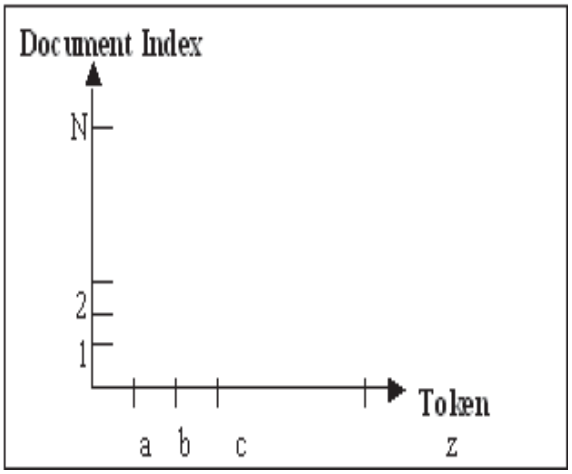

Figure 7. Non-genetic map

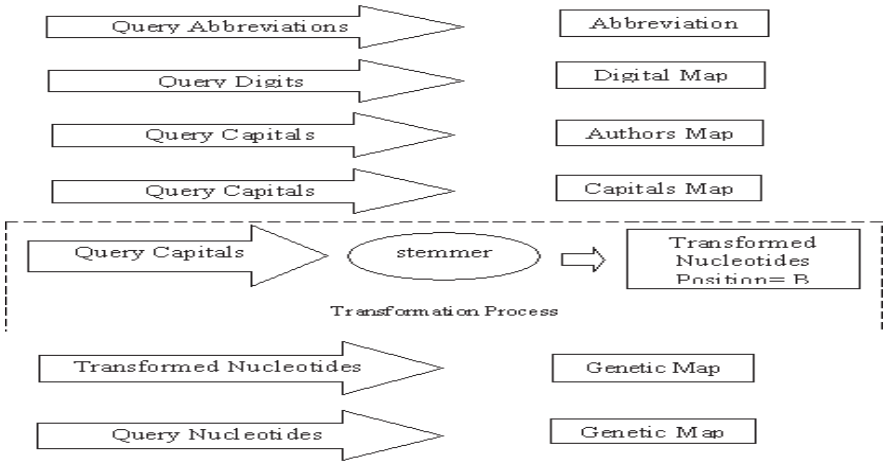

Figure 8. Search Strategy

Equation (4) gives a good equilibrium between tokens of different types in document. However, in most cases we need to differentiate between terms according to their type and term frequencies. In order to do this, we increase the power of the numerator of equation (4) by 0.05, and decrease the power of its denominator by the same value:

$$P_{general}(T_j, D_i) = \frac{(tf_{ij}.Tf_{ij})^{0.25}}{(N_{D_i}.gene\_length_{D_i})^{0.15}} \quad (5)$$

6.3 Position factor

As lower-case token represents the majority of document/ query gene, we may also need to make special discrimination for the members of this category by adding a certain parameter to the power of the numerator, and subtracting it from the power of the denominator. Position factor (P_factor_{ij}), defined below, can play such a role in this aspect.

$$P\_factor_{ij} = \frac{section\_length_i}{2} \times 10^{-1} \quad (6)$$

where $section\_length_i$ is the percentage of nucleotides in each section of the gene.
In this paper, for both documents and queries, we define the $section\_length_i$ as follows:

$$section\_length_i = \begin{cases} 0.7 & for\ Head\_nuc's \\ 0.2 & for\ Body\_nuc's \\ 0.1 & for\ Tail\_nuc's \end{cases} \quad (7)$$

Now, we can define the retrieval status value (RVS) of the biological model to be:

$$RSV_{Biological}(D_i) = \sum_{k=1}^{q} w_{ik}.w_{qk}.idf_j \quad (8)$$

where:

$$w_{ij} = \begin{cases} \dfrac{(tf_{ij}.Tf_{ij})^{0.25}}{(N_j.gene\_length_j)^{0.15}} & for\ non\ genetic\ maps \\ \dfrac{(tf_{ij}.Tf_{ij})^{0.25+P\_factor_{ij}}}{(N_j.gene\_length_j)^{0.15-P\_factor_{ij}}} & for\ genetic\ map \end{cases} \quad (9)$$

$$idf_j = \log_2 \frac{N}{df_j} \quad (10)$$

where:

w_{ik} : term weight of T_k in a document D_i .

w_{qk} : term weight of T_k in a query q.

q: number of search keywords in the query.

Ni: number of documents D_i in the collection.

df_j : number of documents in which term T_j occurs

7. Experiments

In the experiments we used OHSUMED document collections as they were used for the TREC-9 Filtering Track. The OHSUMED collections (ohsu part) provides two lengths for the queries from which we generated two test sets: "ohsu-title" contains the "title" queries (the shortest query representation), and "ohsu-desc" contains the "description" queries. Table 1 shows detailed statistics about test collection.

Table 2 provides tokenization statistics, i.e., statistics obtained from test collections after phase 1 of the biological model. As the table shows, we have five types of tokens for each collection. Nucleotides (number of nucleotides extracted from the documents), capitals, abbreviations, and digits (linguistic tokens extracted from the documents), and authors (statistical tokens of the documents). Table 3 shows statistics of test collection tokens after the mapping phase. It provides each map capacity of tokens after calculating document frequency of each token.

8. Comparisons

Under the same circumstances, we compared our formula with two probabilistic IR models: vector space model (VSM), and Okapi model.

	OHSUMID87	OHSUMID88-91
Number of documents	56,710	293,856
Number of queries	63	63
Average document length	165.06	167.7
Average gene length/document	18.972	19.959
Average gene length/query	4.97 (Title) 5.14 (Description)	4.97 (Title) 5.14 (Description)

Table 1. Statistics about test collections

Type	OHSUMID87	OHSUMID88-91
Nucleotides	456,331	2,466,336
Capitals	119,182	669,494
Abbreviations	51,660	336,925
Digits	221, 685	1,335,350
Authors	189,069	1,056,924
Total	1,037,927	5,865,029

Table 2. Tokenization statistics

Map	OHSUMID87	OHSUMID88-91
Genetic Map	19,724	38,359
Capitals Map	18,265	44,670
Abbreviations Map	6,658	18,050
Digital Map	6,292	15,250
Authors Map	55,554	134,781
Total	106,493	251,110

Table 3. Mapping statistics

(a) Vector Space Model (VSM)

For vector space model, we've used the LNC formula found in (Savoy, Ndarugendamwo and Vrajitoru [13]). According to it, document score is calculated as:

$$RSV_{VSM}(D_i) = \sum_{k=1}^{q} w_{ik}.w_{qk}.idf_j \quad (7)$$

Where $$w_{ij} = \frac{log(tf_{ij})+1}{\sqrt{\sum_{k=1}^{t}(log(tf_{ik})+1)^2}}$$

And

$$idf_j = log_2 \frac{N}{df_j} \quad (8)$$

(b) Okapi Model

To score the relevance of a document versus a query, the following Okapi weighting function is applied (Bertoldi and Federico [1]):

$$RSV_{Okapi}(D_i) = \sum_{k=1}^{q} tf_{qk}.w_{ik}.idf_j$$

where

$$w_{ij} = \frac{tf_{ij} \times (k_1+1)}{k_1(1-b)+k_1 b\frac{\sum_{k=1}^{t} tf_{ik}}{T}+tf_{ij}} \quad (9)(10)$$

$$idf_j = log_2 \frac{N - N_w + 0.5}{N_w + 0.5} \quad (11)$$

where:

tf_{qk} : frequency of term t in query q.

N: number of term occurrences allover the collection.

N_w : number of documents in D which contain term w.

T: average term occurrences allover the collection D.

k_1 = 2, b= 0.6.

Table 4 shows the average precision taken over eleven recall points. It shows that vector space model have an advantage over Okapi model in most places. Usually, if the percentage of change is around (5%), the difference will be considered as significant. Figure 9 shows the recall and precision values obtained from applying our model, the VSM, and Okapi model on the test collections.

	OHSU87 (title)	OHSU87 (desc)	OHSU88-91 (title)	OHSU88-91 desc)
VSM	11.22	20.1	7.5	15.4
Okapi	11.4	19.6	7.05	14.7
Biological Model	13.06	22.9	8.5	16.5
Bio.vs.VSM	+16.5 %	+14.1 %	+14.17 %	+6.8 %
Bio.vs. Okapi	+14.6 %	+17.2 %	+20.89 %	+12.00 %

Table 4. Comparisons: 11-pt Average precision

	OHSU87 (title)	OHSU87 (desc)	OHSU88-91 (title)	OHSU88-91 (desc)
Biological	13.06	22.92	8.5	16.5
After AF help	13.18	23.17	8.16	16.7
Change	+0.91 %	+1.13 %	-4.31 %	+1.41 %

Table 5. Assistant Factor Help: 11-pt Average precision

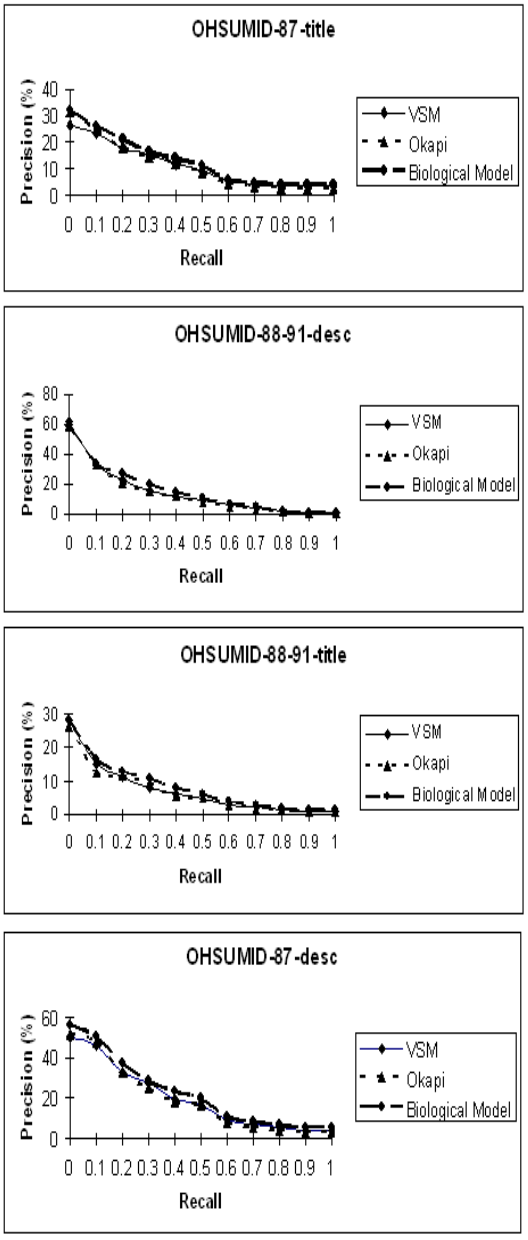

Figure 9. Recall and precision values obtained from applying our model, the VSM, and Okapi model on the test collections.

9. Make Use of Assistant Factor

Until now, nothing is mentioned about how to use the assistant factor to improve the performance of our model. In this section, we'll introduce a new genetic concept; the Identical Nucleotides. Identical nucleotides are those nucleotides that match in both stem and assistant factor in document and query genes. In the following experiment, we'll identify the first 33% of query nucleotides as identical ones, as shown in Figure 10.

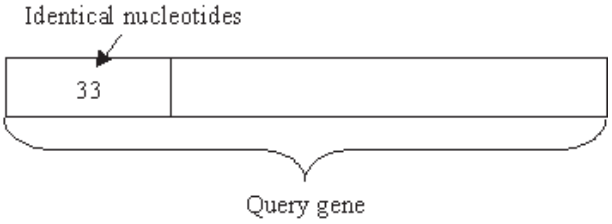

Figure 10. Identical nucleotides within gene query

Now the rule is: A document is eliminated from the retrieved set of documents, if the following conditions are fulfilled:

- the query nucleotide is defined as identical.
- the query nucleotide has a (non-empty) assistant factor.
- the document nucleotide is not a Head one.
- the document nucleotide does not match the query nucleotide identically.

On the other hand, a document score (Doc_Score) is incremented by a certain value which is obtained as follows:

Doc_Score = Doc_Score X (1+2 X P _ factor (Q Nuc)) X (1+ 2 X P _factor (DNuc))

if the following conditions are fulfilled:

- the query nucleotide (QNuc) is defined as identical.
- the query nucleotide has a (non-empty) assistant factor.
- the document nucleotide (DNuc) matches the query nucleotide identically.

After applying this rule on the biological model, we obtained the results shown in Table 5. Although the difference is not significant, it is positive in most cases. Besides, eliminating documents saves considerable time, especially with large collections.

10. Conclusion and Future Work

In this paper, we introduced a new concept for information retrieval modeling that simulates a biological schema. The goal is not just creating a new retrieving formula, but developing a new IR theoretical environment. For the future work, we'll try to improve the performance of the biological model through the genetic map. Also, we will develop a new feedback technique that makes great use of the assistant factor parameter.

References

[1] Bertoldi N., Federico M., (2001). ITC-irst at CLEF 2001 Monolingual and Bilingual Tracks.

[2] Blair D.C., (1990). Language and representation in Information Retrieval, Elsevier, Amsterdam.

[3] Chen H., (1995). Machine Learning for Information Retrieval: Neural networks, symbolic learning, and genetic algorithms, *Journal of the American Society for Information Science*, 46 (3) 194-216.

[4] Croft W. B (2000). Combining Approaches to Information Retrieval, Kluwer Academic Publishers, Boston.

[5] Girgis M. R., Aly A. A., Abdel Latef B. A., El-Gamil B. R., (2006). Biological Environment for Information Retrieval". *In:* Proceeding of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06), p. 1182-1188, Jinan, Shandong, China.

[6] Gordon M., (1988). Probabilistic and genetic algorithms for document retrieval, *Communications of the ACM*, 31 (10) 1208-1218.

[7] Gordon M. (1991). Used-Based Document Clustering by Redescribing Subject Descriptions with a Genetic Algorithm, Journal of the American Society for Information Science, 42 (5) 311-322.

[8] Horng J. T. Yeh C. C., (2000). Applying Genetic Algorithms to Query Optimization in Document Retrieval. *Information Processing and Management*, 30, 753-759.

[9] Hust A., Klink S., Junker M. Dengel A (2002). Towards Collaborative Information Retrieval: Three Approaches, Text Mining — Theoretical Aspects and Applications, Physica-Verlag.

[10] Petry F., Bucklws B., Prabhu D., Kraft D (1993). Fussy information retrieval using genetic algorithms and relevance feedback, *In:* Proceeding of the ASIS annual meeting, p. 122-125.

[12] Raghavan V.V., Agarwal B., (1987). Optimal determination of user-oriented clusters: An application for the reproductive plan, *In:* Proceedings of the second conference on genetic algorithms and their applications, Hillsdale, NJ, pp. 241-246.

[13] Salton G., McGill M. J., (1983) "Introduction to Modern Information Retrieval", McGraw-Hill (NY).

[14] Savoy J., Ndarugendamwo M., Vrajitoru D., (1996). Report on the TREC-4 Experiment: Combining Probabilistic and Vector-Space Schemes, *In*: Proceedings TREC'4 , NIST, Gaithersburg (MD), October 1995, p. 537-547.

[15] Song F., Croft W. B., (1999). A General Language Model for Information Retrieval, *In*:Proceedings of Eighth International Conference on Information and Knowledge Management, Kansas City, MO, November 2-6.

[16] Van Rijsbergen, C., (1979). Information Retrieval, Butterworths, London.

[17] Vrajitoru D., (1998). Crossover Improvement for the Genetic Algorithm in Information Retrieval, *Information Processing and Management* , 34 (4) 405-415.

[18] Vrajitoru D., (2000) "Large Population or Many Generations for Genetic Algorithms? Implications in Information Retrieval", In F. Crestani, G. Pasi (eds.): Soft Computing in Information Retrieval. Techniques and Applications, Physica-Verlag, Heidelberg, pp. 199-222.

[19] Yang, J. J., Korfhage, R. R., Rasmussen, E (1992). Query Improvement in Information Retrieval Using Genetic Algorithms, *Proceeding of the ACGA*, p. 603-611.