

**Evolutionary Computation in Conceptual Clustering
and Tagging**

Takaoki Ueda
IlliGAL Report No. 2008012
May, 2008

Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue Urbana, IL 61801
Office: (217) 333-2346
Fax: (217) 244-5705

This thesis is submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science in the College of Engineering of the University of Illinois at Urbana-Champaign, 2008.

Abstract

The Web 2.0 technologies provide users with collaborative work-spaces over the Internet. For example, Wikipedia is an open source encyclopedia that anyone can edit articles. YouTube provides spaces where users can share videos and annotations about them. Users can put images on Flickers and collaborate each other by categorizing with tagging. These contents are created by users' voluntary activities, which is one of the features for the Web 2.0 technology. Some services based on the Web 2.0 have well organized text-based contents on them. For example, Wikipedia has well structured contents, which is due to voluntary activities of the users trying to edit each contents so as to be sophisticated. On the other hands, other services, such as YouTube and Flickers', only have short sentences or small number of words as annotations. Additionally these annotations are usually different according to each user because participants are not in situation of collaborations. As a result, annotations do not have meaning for videos and pictures. A system that converts annotations into meaningful texts is useful because building texts requires resources while a number of annotations are available.

The purpose of this thesis is development of the text builder which is based on the Web 2.0 technology with genetic algorithms and natural language processing. A human interactions system is developed in this thesis for automatically building meaningful tags from annotations. The system consists of mainly two parts: a conceptual clustering component based on natural language processing and a sentence creating component based on genetic algorithms. The conceptual clustering component decomposes annotations into phrases with main ideas. Then, the sentence creating component builds several tags from the phrases. Thirdly created tags are evaluated by users to find better explanations for videos and pictures. Participants are supposed to collaborate through evaluations to create more organized and meaningful tags.

The developed system succeed in creating tags from annotations without structures through user-machine interactions. This system is applicable to other systems which has only annotations as participants' comments. Because tags created by this system have meanings but short length a system building longer text as sentences is left as future works.

Chapter 1

Introduction

Due to the rapid growth of the Internet, human interactions on the web have exponentially increased. The Web 2.0 technology utilizes and simultaneously enhances human interaction. At the same time computational models for knowledge generation are also available. However there is a small number of applications using a combination of them; it is expected to provide more sophisticated knowledge through human and machine interactions.

The purpose of this thesis is the development of a collaborative tag builder which has human-machine interactions based on the Web 2.0 technology, genetic algorithms, and natural language processing. A complete literature review of the computational models, natural language processing, information retrieval and machine learning can be found elsewhere (Walker, Rambow, & Rogati, 2001; Manning, Raghavan, & Schütze, 2008; Mitchell, 1997).

Natural language processing (NLP) is the technique for understanding natural languages with computational models. There are a number of applications. For example, the textual entailment system (Braz, Girju, Punyakanok, Roth, & Sammons, 2006) creates a shorter sentence without changing the meaning for given text. The question answering system (Moldovan, Harabagiu, Girju, Morarescu, Lacatusu, Novischi, Badulescu, & Bolohan, 2002) generates answers corresponding to given question sentences. The natural language generation generates sentences based on the grammar framework (Walker, Rambow, & Rogati, 2001). NLP is used to find appropriate synonyms in the system proposed in this thesis.

Information retrieval (IR) is an important research area within computer science. Constructing search systems is one of the objectives in IR. The concept of information retrieval is quite old (Bush, 1945). IR methods also capture text structures. For example, a word model is defined by Bayesian network (Ponte & Croft, 1998). Systems for tagging, phasing and speech recognition have been proposed (Song & Croft, 1999). For further understanding, refer to *Introduction of Information Retrieval* (Manning, Raghavan, & Schütze, 2008).

Machine learning (ML) is a subfield of artificial intelligence. It focuses on developing algorithms that enable computers to learn the objective functions. ML has been applied to support NLP and IR as the optimization tools (Braz, Girju, Punyakanok, Roth, & Sammons, 2006; Ponte & Croft, 1998). Genetic algorithms are also used in ML. In general, there are two kinds of ML. One is supervised learning method which learns the objective functions from training data set. The other one is unsupervised learning methods without using training data set. *Machine Learning* (Mitchell, 1997) is a useful resource for better understanding the idea of ML.

On the other hand, Web 2.0 provides new collaborative spaces on the Internet. The key feature of Web 2.0 is that participants share information voluntarily. For example, social network services such as Facebook and MySpace supply users with spaces where they can exchange their information. In other services, such as YouTube and Flickr's, users can share videos and pictures and leave annotations about the contents. Although the annotations are not well structured, and each one is based on subjective views, their number is numerous. Therefore a system which converts annotations into meaningful tags and descriptions with structure is useful. Since combinations of human interactions with Web 2.0 and computational models are expected to perform better; a system with the combinations to build meaningful tags is proposed in this thesis.

The developed system is a web-based system that stores pictures and their annotations. The system then processes stored annotations to build tags that are evaluated by users. There are two main components in this system: a conceptual clustering component from plain documents and a tag building component. The former extracts the main idea of annotations, and the later suggests most suitable tags for pictures.

The purpose of this thesis is twofold:

- **Extract key concepts of annotations:** This is a preprocessing twofold step to build tags. Computational modeling methods are used to obtain main ideas from user's annotations.
- **Apply GA operations for building tags:** This is a phase for human and machine interactions to combine human knowledge and computer generated knowledge.

The thesis is structured as follows. Chapter 2 gives a brief introduction to genetic algorithms and its variations, such as the interactive GA and the human-based GA. The potential of GAs for integrating human and computer generated knowledge is discussed. Chapter 3 describes conceptual clustering, the method for extracting key idea of annotations as building blocks (Ueda, Llorà, Goldberg, Yasui, & Sastry, 2007). Chapter 4 proposes a picture tagging system. This system uses a genetic algorithm to improve tags assigned to pictures by users. The experimental results with the prototype system are reported in this chapter. Finally chapter 5 summarizes the work presented and outlines future work.

Chapter 2

Introduction to Genetic Algorithms

This chapter presents a brief introduction to genetic algorithms and how they, when coupled together with a tag builder can help integrate user interactions and computational models. The rest of this chapter presents: (1) a brief introduction to genetic algorithm, (2) interactive and human-based genetic algorithms, (3) genetic algorithm for collaborative editing, and (4) possible benefits of using *competent* genetic algorithm as part of the tag builder.

2.1 Introduction of Genetic Algorithms

This section briefly introduces genetic algorithms. GAs are search, optimization and machine learning methods based on the principles of natural selection and genetics (Holland, 1975; Goldberg, 1989; Goldberg, 2002). Analogous to the natural operations, GAs combine the survival of the fittest among strings with the innovative flair of human search. The strings are the candidate solutions and they are called *chromosomes*. The characters of features in chromosomes are described as *genes*. The values of genes are referred to as *alleles*. The relative measurement of the candidate solution is described as *fitness*. The set of collection of candidate strings, the *population*, is also an important concept (Goldberg, 1989). Since GAs imitate the natural evolution of genes, the population size has an affect on its performances. This is a part of difference between GA and the other searching methods. The idea of population enables GA to handle noise, rugged landscapes, and bad scaling (Sastry, 2007). An early example of GA implementation is the simple genetic algorithm (sGA) (Goldberg, 1989)

2.1.1 The Simple Genetic Algorithm

The simple GA does not involve any complex operations other than copying chromosomes and swapping or changing the parts of them. The reasons why this simple mechanics works is both subtle and strong. Simplicity of operations and strength of effect are the main attractions of the genetic algorithm approach.

A simple genetic algorithm that produces good outcomes is composed of three operations:

Reproduction is the process of duplicating individual strings according to their objective function values.

Duplicating according to their fitness values makes that strings with the higher value have contribute one or more offspring with higher probability in the next generation.

Recombination is the process of producing next generation populations through crossover and mutation.

Parents chromosomes are selected for each child from the mating pool. New solutions are created which usually shares many of the traits of their parents.

Mutation is a process of creating new identical strings except few changes of traits. Making just few changes means performing a random walk in the vicinity of a candidate solution.

For a more detailed discussion see (Goldberg, 1989; Goldberg, 2002).

2.2 Interactive and Human-Based Genetic Algorithms

A collaborative tag builder integrates user knowledge through interactions between users and computers. This chapter explains two kinds of GAs that can import the ideas from users to the system for the innovation.

2.2.1 Interactive Genetic Algorithm

In a simple GA, the fitness is often computed based on an objective function or functions. There are, however, some problems that are difficult to evaluate in a computational procedure. If the fitness function only exists in the mind of the user, evaluating candidate strings as a computational procedure is not possible. An idea of outsourcing to human a evaluator could solve these difficulties. For example, Caldwell (1991) show how the reconstruction of a criminal’s face could be done following such methodology. A genetic algorithm operating all of the operations except evaluating the fitness of candidate faces; the witness evaluates the faces based on his/her memory (Caldwell & Johnston, 1991). This is called the interactive genetic algorithm. The interactive GA has been achieved wide-range of successes (Takagi, 2001).

2.2.2 Human-Based Genetic Algorithm

Whereas interactive GAs fitness outsources the evaluation process to human agents, other parts are operated via computational procedure. On the other hand, human-based genetic algorithms also outsource the recombination step (Kosorukoff, 2001). Human-based GAs seeks to encourage the creative potential of users involved in the process. The human-based GAs have been applied to the focus group discussion for idea creation (Goldberg, 1993; Goldberg, D.E., Welge, M., & Llorá, X., 2003; Saruwatari, Llorà, Goldberg, Yasui, Sastry, & Hiroshi, 2008).

2.3 Genetic Algorithm for Tag Builder

This section discusses why GAs can integrate both human-generated and computer-generated knowledge. The interactive and the human-based GAs bridge together users and computers. Any operators which can not be processed on computers could be outsourced. Therefore, GAs enable tag builder to manage human-machine interactions.

Via outsourcing operations, genetic algorithms handle difficult natural language problems like collaborative tagging. Since NLP still requires human supports to deal with the complexity of natural language, a GA is adapted to utilize its human machine interaction ability. To create a collaborative tagger requires two steps: (1) identify three operations whether each of them should be a outsourced procedure or a computational procedure; (2) design a *competent* GA through identifying building blocks (BBs). One definition of a BB is the “highly fit short-defining-length schemata” (Goldberg, 1989) and more elaborate definition is explained elsewhere (Goldberg, 2002). Among candidates, there are certain schema which lead to high fitness. The sequence of figuring out and recombining BBs during GA operations is necessary.

Chapter 3

Extracting Conceptual Clusters of Documents

This chapter describes the conceptual clustering method for extracting conceptual clusters from annotations. As discussed in chapter 1, constructing tag builder needs to combine the computational models and interactions of users. Therefore, it is important to develop computational models which can be integrated with the human-generated knowledge. This chapter introduce conceptual clustering of texts, as the part of the computational models of collaborative tag. Since tag builder needs to create tags quickly, reliably, and accurately through applying a GA, the step toward designing *competent* GA is necessary, which involves building block (BB) discoveries as shown in chapter 2.

3.1 Overview of Conceptual Clustering as Building Block Discovery

Here, a brief roadmap of the conceptual clustering method is given. As the first step toward BB discovery, the assumption is the BBs of NLP is the sets of words. The goal is to identify sets of words from given text stream that in particular contexts form semantic unity. In this section, we present a brief overview of the overall approach.

The approach consists of three processes:

1. Text processing of text streams
2. Semantic group identification
3. Building block discovery

For example, consider the following document, where each paragraph is identified by a unique number. All of these paragraphs have similar meaning when used in the same context.

1. This is a long story which cannot be described in one paragraph. However, we can describe this story by repeating paragraphs with the same meaning many times.
2. This story is ambiguous and huge. Therefore, we cannot write it in one paragraph. One way to describe this story is to repeat paragraphs using similar words.
3. There is a story which cannot be described in one paragraph. However, we can write this paragraph repeatedly to describe it. Repeating is the only acceptable solution towards expression.
4. This is a story which cannot be expressed in one paragraph. There is one way to describe it and that is by repeating paragraphs with the same meaning as this paragraph many times.
5. One paragraph is not sufficient to express our story. By writing paragraphs with same meaning many times, it is possible to express our story.
6. Without repeating paragraphs with the same meaning using similar words, we can not describe our story.

Text Processing: As a first step, we need to clean the document to ease the later processes. This step includes stop word elimination and word stemming. The former is to eliminate words irrelevant to the document context, and the latter is to normalize each word to its base form. For example, this step converts the paragraph 1 in the sample document above to ‘‘story describe one paragraph. describe story repeat paragraph mean time’’.

Semantic Groups Identification: A semantic group is a cluster of words with similar meaning. Similarities between words are measured using *hypernyms*. A *hypernym* is a word that is more generic than a given word. Our experiments use WordNet for obtaining hypernyms of each word. Note that every word in WordNet (Miller, G., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K., 1990) has at least one hypernym, that is, *entity*. For example, you can find semantic groups obtained from the sample document in the Section 0.8. Identified semantic groups are indexed by a word in the group.

Context-aware Semantic Building Block Identification: In the previous step, we have a set of words (including index words to semantic groups). The last step is to identify building blocks from the set of words. For example, as we report in the Section 0.8, obtained building blocks from the sample document were ‘‘one paragraph’’ and ‘‘describe repeating’’.

3.2 Text Processing

Text processing is executed prior to the other processes, in order to reduce the ambiguousness of the text and help to create a compact word set and facilitate later processes. This section describes three types of text processing used in our approach: stop words elimination and stemming.

Stop Words Elimination

A *stop word* is a word which is commonly and frequently used in texts. It is filtered out from texts, because typically, it is irrelevant to the essence of the text. Our stop word list includes pronouns, prepositions, articles, adverbs and some verbs (Weiss, Indurkha, Zhang, & Damerau, 2005).

Stemming

Stemming is a process for normalizing variants of words by removing morphological and inflectional endings from the words. For example, by the stemming process, all of the words *remove*, *removes*, *removed*, and *removing* are converted into *remov*. Our stemming process uses the Porter stemming algorithm (Porter, 1980), which is a widely used stemming algorithm for English. The algorithm detail is beyond the scope of this paper. Stemmed words are used internally to further processes, and original words are used to show results.

Splitting

Splitting decomposes a document into paragraphs. The splitting is done by a heuristic way based on a assumption that each paragraph (1) has more than 100 characters and (2) ends dot (.) and tab (\t) or dot (.), space (), and tab (\t).

3.3 Semantic Group Identification

A semantic group indicates a cluster of words with similar meanings. We use a design structure matrix (DSM) clustering method for identifying the semantic group. In this section, first we give a brief review of DSM clustering method. Next, the approach for identifying semantic groups by DSM clustering method is proposed.

3.3.1 DSM Clustering

A dependency structure matrix (DSM) is essentially an adjacency matrix representation of a graph where each entry d_{ij} represents the dependency between node i and node j (Steward, 1981; Yassine, Falkenburg, & Chelst, 1999). Entries d_{ij} can be real numbers or integers. The larger the d_{ij} is, the higher the interaction is between node i and node j . If we focus on the 0-1 domain, this then $d_{ij} = 0$ means that node i and node

j do not interact, and $d_{ij} = 1$ means that node i and node j interact with each other. The diagonal entries (d_{ii}) have no significance and are usually set to zero or blacked-out. For elaborate exposition of DSM, please see MIT DSM web site: <http://www.dsmweb.org/>.

The goal of DSM clustering is to find subsets of DSM elements (*i.e.*, clusters) so that nodes within a cluster are maximally interacting, and clusters are minimally interacting. In a typical DSM clustering problem, overlapping clusters (clusters that share same nodes) are permissible.

Our approach uses DSM clustering (Yu, Yassine, & Goldberg, 2005) metric based on the minimal description length principle (MDL) (Rissanen, 1978). Previous DSM clustering algorithms can be found elsewhere (Fernandez, 1998; Sharman, Yassine, & Carlile, 2002). Their results showed that the objective function used was short of accurately predicting “good” clustering because of the oversimplified objective function.

Suppose that we have a model which describes a given data set, $DSM = [d_{ij}]$. Here, the model means a description that specifies which node belongs to which cluster. Usually, the model does not completely describe the given data; otherwise, the model would be too complex to use. Therefore, the description length that the model needs to describe the given data consists of two parts: (1) the model description and (2) the mismatched data description. This scheme may be easier to understand in light of the following sender-receiver example.

Assume a sender has a given data set which is needed by the receiver. Given a model that approximately describes the given data set, the sender first sends the model (*i.e.*, model description) to the receiver. To ensure that the receiver gets exactly the same data set, the sender is also required to send the data which is mis-described (*i.e.*, mismatched data description) by the model sent earlier. If the model is too simple, the model description is short: however many data mismatches exist and the mismatched data description becomes longer. On the other hand, a complicated model reduces mismatched data, but the model description is longer.

The minimum description length principle (MDL) (Rissanen, 1978) satisfies our needs for dealing with the above trade-off. The MDL can be interpreted as follows: among all possible models, choose the model that uses the minimal length for describing a given data set (that is, model description length plus mismatched data description length). There are two key points that should be noted when MDL is used: (1) the encoding should be uniquely decodable, and (2) the length of encoding should reflect the complexity. For example, the encoding of a complicated model should be longer than that of a simple model. Next, we define the MDL clustering metric in detail.

Model Encoding. The way we encode the model is straightforward. The description of each cluster starts with a number which is sequentially assigned to each cluster, and then this is followed by a sequence of nodes in the cluster. It is easily seen that the length of this model description is as follows:

$$\sum_{i=1}^{n_c} (\log_2 n_n + cl_i \cdot \log_2 n_n), \quad (3.1)$$

where n_c is the number of clusters in the model, n_n is the number of nodes, cl_i is the number of nodes in the i -th cluster.

Mismatched Data Description. Based on the model, we first construct another DSM (call it $DSM' = [d'_{ij}]$), where each entry d'_{ij} is 1 if and only if some cluster contains both node i and node j simultaneously.

Then, we compare DSM' with the given DSM . For every mismatched entry, where $d'_{ij} \neq d_{ij}$, we need a description to indicate where the mismatch occurred (i and j) and one additional bit to indicate whether the mismatch is zero-to-one or one-to-zero. Define a mismatch set $S = \{(i, j) | d'_{ij} \neq d_{ij}\}$. The mismatched data description length is given by:

$$\sum_{(i,j) \in S} (\log n_n + \log n_n + 1). \quad (3.2)$$

The first $\log n_n$ in the bracket indicates i , the second one indicates j , and the additional one bit indicates the type of mismatch.

The MDL clustering metric is given by the summation of the model description length and the mismatched data description. With some arithmetic manipulations, the metric can be expressed as follows:

$$f_{DSM}(M) = \log n_n \sum_{i=1}^{n_c} (cl_i + 1) + |S|(2 \log n_n + 1), \quad (3.3)$$

where n_c is the number of clusters, n_n is the number of nodes in the DSM, cl_i is the size of the i -th cluster, and S is a mismatch set.

With the above metric, the DSM clustering problem is converted into an optimization problem: Given a DSM, the objective is to find a DSM clustering arrangement (model, M) to minimize the above metric (f_{DSM}).

3.3.2 Semantic Groups Identification by DSM Clustering

Semantic groups are identified by the DSM clustering method reviewed above. DSM clustering is used mainly because of its ability to infer overlapping clusters—a common property of text where one word may have different meanings and, hence, belong to different groups. This subsection focuses on how to create a DSM from a set of words.

Let W be a set of words contained in texts. Hypernym induced vectors $H(w_i)$ and $H(w_j)$ for $w_i \in W$ and $w_j \in W$ are obtained by hypernym sets of w_i and w_j . Let H_{w_i} and H_{w_j} be hypernym sets of word w_i and word w_j , respectively. Let $H = H_{w_i} \cup H_{w_j} = \{h_1, h_2, \dots, h_n\}$ denote a combined set of hypernyms derived from w_i and w_j . $H(w_i)$ and $H(w_j)$ are defined by

$$H(w_i) = \{H_{w_i}(h_1), H_{w_i}(h_2), \dots, H_{w_i}(h_n)\} \quad (3.4)$$

$$H(w_j) = \{H_{w_j}(h_1), H_{w_j}(h_2), \dots, H_{w_j}(h_n)\}, \quad (3.5)$$

where

$$H_{w_i}(h_k) = \begin{cases} 0 & \text{if } h_k \notin H_{w_i} \\ 1 & \text{if } h_k \in H_{w_i} \end{cases} \quad (3.6)$$

Likewise, $H_{w_j}(h_k)$ is 0 (if $h_k \notin H_{w_j}$) or 1 (if $h_k \in H_{w_j}$).

Let $\theta(w_i, w_j)$ be similarity between w_i , and w_j . $\theta(w_i, w_j)$ is measured by cosine similarity of hypernym induced vectors, as follows.

$$\theta(w_i, w_j) = \arccos \left[\frac{(H(w_i))^T \cdot H(w_j)}{\|H(w_i)\| \|H(w_j)\|} \right], \quad (3.7)$$

Let D denote a DSM for identifying semantic BBs is a n -square matrix, where $n = |W|$ (i, j) entry ($i \neq j$) of D is defined as follows.

$$D(i, j) = \begin{cases} 0 & \text{if } \theta(w_i, w_j) < \delta \\ 1 & \text{if } \theta(w_i, w_j) \geq \delta \end{cases} \quad (3.8)$$

where $w_i, w_j \in W$ and δ is a threshold. Intuitively, the entry is 0, if the corresponding two words have similar meaning, and vice versa. In our experiment, $\delta = 1.10$.

3.3.3 Simple Example

Assume we have a set of words $W = \{dog, cat, potato, rat\}$ derived from a given text. For example, $\theta(dog, cat) = 1.13$, then (dog,cat) entry of DSM = 1. Similarly, obtaining θ for every pair of words, DSM for W is represented as Table1.

Table 3.1: DSM matrix.

-	dog	cat	potato	rat
dog	x	1	0	1
cat	1	x	0	1
potato	0	0	x	0
rat	1	1	0	x

Table 2 shows the reordered matrix obtained by DSM clustering.

From this matrix, we have a cluster with “dog”, “cat” and “rat” and a singleton “potato”.

Table 3.2: Reordered DSM matrix.

-	dog	cat	rat	potato
dog	x	1	1	0
cat	1	x	1	0
rat	1	1	x	0
potato	0	0	0	x

3.4 Context-aware Semantic Building Blocks Identification

A context-aware semantic building block indicates a cluster of words having close correlations, that is, similar semantic content of the given document. The BBs are identified by the extended compact genetic algorithm (eCGA). In this section, first we briefly review the eCGA, and then describe our approach to use the eCGA on our problem.

3.4.1 Extended Compact Genetic Algorithm and Minimum Description Length

The extended Compact genetic algorithm (eCGA) is a probabilistic algorithm which was developed to learn linkage through correcting good probability distributions (Harik, Lobo, & Sastry, 1999). The measure of desired distribution is quantified based on minimum description length (MDL) model. A class of probability model, marginal product models (MPMs), is used in eCGA for a probability distribution.

The key concept of the MDL model is as follows. Simpler distributions are better than the more intricate one, and everything is equal. The MDL restriction penalizes both inaccurate and complex models, therefore conducting to an optimal probability distribution. MPMs comprise of a product of marginal distributions on a partition of the genes. MPMs also make easier a direct linkage map with each partition separating tightly linked genes.

The eCGA has consecutive sequences as follows.

1. Generate a random population of size N .
2. Undergo tournament selection at rate S .
3. Model the population using a greedy MPM search.
4. If the model has converged, stop.
5. Generate a new population using the given model.
6. Return to step 2.

Two things need further explanation: (1) the determination of MPM using MDL, and (2) the generation of a new population based on MPM. A constrained optimization problem determines MPM in every generation. The determination of MPM in each generation can be represented as minimizing the sum of the model complexity (C_m) and the cost of using single model compare with complex one (C_p). Model complexity C_m and cost of using single model C_p can be represented as follows.

$$C_m = \log_2(n) \sum_{i=1}^m (2^{k_i} - 1) \quad (3.9)$$

$$C_p = \sum_{i=1}^m \sum_{j=1}^{2^{k_i}} N_{ij} \log_2 \left(\frac{n}{N_{ij}} \right) \quad (3.10)$$

where n is number of populations, m in the equations represent the number of BBs, k_i is the length of BB $i \in [1, m]$, and N_{ij} is the number of chromosomes in the current population possessing bit-sequence $j \in [1, 2^{k_i}]$ population size.

The greedy MPM search used in eCGA begin with a model that all the variables are independent. Eventually, independent variables merge into subsets till the MDL model no longer progresses. The methodology of generating a new population is as follows: a new population of $n(1 - p_c)$ where p_c is the crossover probability are the best individuals in the current population. The others, $n \cdot p_c$, are filled with randomly chosen subsets from the current individuals based on the probabilities of the subsets.

One of the most important parameters of eCGA is the required population size. The analytical population sizing method is developed elsewhere (Sastry & Goldberg, 2004). This analysis estimates the required population size as follows:

$$n \propto 2^k \left(\frac{\sigma_{BB}^2}{d^2} \right) m \log m, \quad (3.11)$$

where m is number of building blocks, k is the building block length, and $\frac{\sigma_{BB}^2}{d^2}$ is the noise-to-signal ratio (Goldberg, Deb, & Clark, 1992).

In this subsection, we reviewed how BBs are created in eCGA effectively. The description for NLP is described in following subsections.

3.4.2 Semantic BB Identification by eCGA

Semantic BBs are identified by the eCGA method. eCGA is mainly used because of its ability to minimize complexity among BBs. eCGA requires vectors as inputs. Suppose a given document consists of paragraphs. Let D be the document which is a set of paragraphs, and denote each paragraph by P_i , that is, $D = \{P_1, P_2, \dots, P_n\}$. Each paragraph P_i is converted into a vector V_i for using eCGA as follows.

Let W be a set of semantic groups obtained from the document and other words (that is, not clustered words). Paragraph vector V_i is defined by

$$V_i = (P_i(w_1), P_i(w_2), \dots, P_i(w_n)) \quad (3.12)$$

where $w_k \in W$, $1 \leq k \leq n$ and

$$P_i(w_k) = \begin{cases} 0 & \text{if } w_k \text{ is not contained in } P_i \\ 1 & \text{if } w_k \text{ is contained in } P_i \end{cases} \quad (3.13)$$

This creates a population form from each paragraph. Each binary variable represents the existence of semantic group set in the paragraph. The final step is using eCGA to obtain context-aware semantic building blocks. The building blocks are identified by the MPM. MPM is provided by the model-building process of eCGA.

3.5 BB discovery experiments

For examining our approach, we performed experiments using two documents: (1) a sample document consisting of six paragraphs presented in Section 0.4, and (2) a news article set obtained from Google News on January 29th, 2007. In this section we report the experimental results from each document and show how our approach identifies BBs from the documents.

3.5.1 Experiment 1

As a preliminary experiment, we examined our approach to a small sized sample document, which was shown in Section 0.4. The sample document consists of six paragraphs.

After the first step of our approach (stop words elimination and stemming words), DSM clustering method identified six semantic groups (see Table 0.8.1). Each row indicates an index word (left column) of a word group (right column). Each pair (or triplet) has common hypernyms. For example, all of the words “possible”, “meaning” and “one” have a hypernym “cognition” in common.

After clustering the semantic groups provided by DSM, we have a set of 36 semantic groups. Our approach identified two BBs having two words [one paragraph][describe repeating]. The example paragraphs uses either “describe” or “write”. Only one paragraph uses both, but the semantic layer brought them together. This shows clustering semantic BBs raises efficiency and quality of BBs.

Table 3.3: Experiment 1: Semantic groups

Index word	group
describe	[describe write]
meaning	[possible meaning]
one	[one meaning]
solution	[solution meaning words]
words	[words way]
closer	[closer possible]

3.5.2 Experiment 2

To validate our approach, we performed an experiment using a larger document. The document consists of around a hundred news articles about *a suicide bombing in Israel* obtained from Google News on Jan 29th, 2007.

The stop words elimination reduced the size of word set to half approximately (from 1224 words to 643 words). Then, DSM clustering identified the various sizes of semantic groups. Figure 1 shows the number of identified semantic BBs and the sizes of groups (i.e., number of words contained in the group). DSM clustering identified 168 semantic groups. The average size of the semantic groups was approximately 5.28.

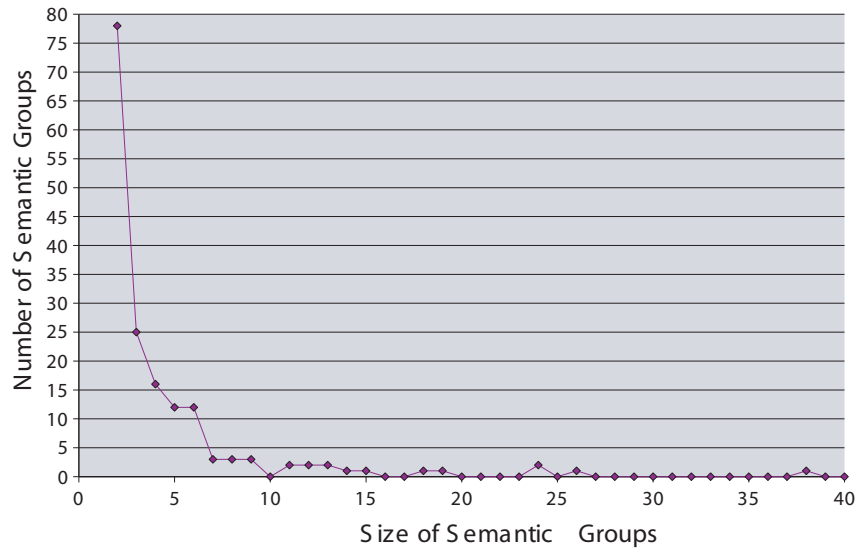


Figure 3.1: Distribution of Semantic Group Size

After DSM clustering step, we have a set of 283 semantic groups identified from the 608 paragraphs available. Table 0.8.2 shows the number of identified BBs of each size. The average size of BBs was approximately 1.74.

Table 3.4: # of BBs and its size

Size of BBs	1	2	3	4
# of BBs	69	69	24	1

Table 5 shows the top ranked BBs for each size.

Table 3.5: Examples of identified BBs

Size	BBs
4	[suicide israel april restaurant]
3	[agree occupation march], [gaza strip city], [violence undermine stage], [west bank military], [resort sea tip]
2	[brother thursday], [jihad joint], [iran february], [crackdown fire], [lead ramadan], [exile iranian]

Each BBs in table 5 actually represents the semantic context in the document. In other words, we can have an intuitive understanding of the article only by reading the generated BBs. These BBs are well suited for HBGA operation.

3.6 Summary

This chapter described the conceptual clustering methods as a BB discovery problem, such as, eliminating noise words; creating the temporary dictionary to convert the words into the vectors; identifying the words which should be recognized as the similar; clustering the similar words group by DSMGA which will be registered into the dictionary as one word; creating the dictionary by using these groups; and splitting BBs by using ECGA. Since the conceptual clusters are extracted as BBs, this could extend to design a *competent* GA to apply a tag builder. In the next chapter, this thesis explains the tagging system for pictures as a example of the integration of user interactions.

Chapter 4

Picture Tagging System

As shown in the chapter 1 and 2, collaborative tagging system requires human interactions. The interactive GA and the HBGA can be used for integrating interactions between computer and users.

Picture tagging system is common nowadays. Tags assigned to pictures mostly rely on users' subjective view of the pictures. The users' view is often different from person to person and changing from time to time. The tags someone assigned sometimes don't make any sense to the others. In addition, if users didn't know what it is on the picture, he/she would never input any tags.

This chapter focuses on picture tagging system. This system automatically enhances quality of tags based on users' evaluations. After a methodology for the system is proposed, the overview of the prototype system is presented.

The structure of this chapter is as follows:

1. methodology
2. prototype system overview
3. empirical study
4. future work

Each of these is discussed in the remainder.

4.1 Methodology

This section discusses how a GA can be applied for enhancing quality of tags assigned to pictures. In this system, tags are considered as chromosomes. The words consisting of tags are considered as genes.

4.1.1 Initialization

To apply a genetic algorithm, an initial population must be prepared. Traditionally, the initial population is generated randomly. However, the random generation of tags involves significant difficulties due to the nature of natural language.

This system employs the tags assigned by users for the initial population.

4.1.2 Fitness

The fitness for each tag on a picture is determined based on a user evaluation. Each tag has a point score p , which is initially $p = 0$. The point score p is increased one by user's positive evaluation, that is done by clicking "POS". Conversely, the point score p is decreased one by user's negative evaluation, that is done by clicking "NEG". Let n be a number of accesses to the picture. Then the fitness of each tag is calculated by p/n .

4.1.3 Operators

As shown in Chapter 2, mutation, selection and recombination are three operators in GAs. Currently, mutation and selection are implemented in this system. Recombination is left for future work.

Mutation

As shown in Chapter 2, mutation performs the random walk in the neighborhood of candidate solutions (Goldberg, 2002). In the proposed tagging system, the mutation operation replaces randomly selected words in a tag with their synonyms. The synonyms are obtained from the WordNet dictionary.

First, the part of speech tagger (Loper & Bird, 2002) gives *POS labels* to all words in the tags. POS label shows the part of speech for each word. Next, random number between 0 to 1 is assigned to each word in a tag. The words with the random number less than 0.3 are selected as the words for mutation. Let $W = \{w_1, w_2, \dots, w_n\}$ be the set of selected words.

For each $w_i \in W$, a synonym set $S(w_i) = \{s_{i1}, s_{i2}, \dots, s_{im}\}$ is obtained by the WordNet dictionary. Note that all $s_{ij} \in S(w_i)$ have same POS labels as w_i .

Let $Sim(w_i, S(w_i)) = \{sim(w_i, s_{i1}), sim(w_i, s_{i2}), \dots, sim(w_i, s_{im})\}$ be a vector of similarity values between the selected word w_i in W and its synonym $s_{ij} \in S(w_i)$. Similarity between the word w_i and s_{ij} is obtained by using a function of the natural language toolkit (Loper & Bird, 2002). Each $sim(w_i, s_{ij}) \in Sim(w_i, S(w_i))$ is normalized to $\sum_{1 \leq j \leq m} sim(w_i, s_{ij}) = 1$.

This mutation operator replace the word w_i with its synonyms s_{ij} . s_{ij} is selected with probability proportionally to the $sim(w_i, s_{ij})$. The higher $sim(w_i, s_{ij})$, the probability that s_{ij} is selected becomes the higher.

Selection

The elitist scheme is employed for selection operator, which is different from the reproduction operator mentioned in Chapter 2. The unselected tags are stored as just one chromosome in the population. Although the system maintains both selected and unselected tags, only 10 of them with higher fitness will be displayed and evaluated by users. This system has to adapt the different selection operator because human can evaluate small number of tags. Otherwise, the populations converge rapidly, so our objective which is to make the most suitable tags can not be achieved.

4.2 Prototype System Overview

In order to evaluate proposed methodology, a web based collaborative tagging system called *picture descriptor* has been developed.¹

The picture descriptor is developed in Python with the natural language toolkit (Loper & Bird, 2002), and has the following functions:

1. Uploading pictures: By clicking the *Upload new picture* button ((1) in Figure 0.11), users can upload pictures. Users must evaluate at least two pictures before they upload their pictures. This rule assures that the system has both evaluations and tags many enough.
2. Adding tag to each picture: By sending text through the text field ((2) in Figure 0.11), users can add tags to each picture. There is no limitation on the number of tags user can add to each picture.
3. Evaluating the tag for each picture: By clicking “POS” or “NEG” buttons attached to each tag ((3) in Figure 0.11), users can evaluate whether the tag is appropriate (“POS”) or not (“NEG”). Each user can evaluate a maximum six tags for each picture.

¹ <http://www.communusphere.com/>

Picture Descriptor

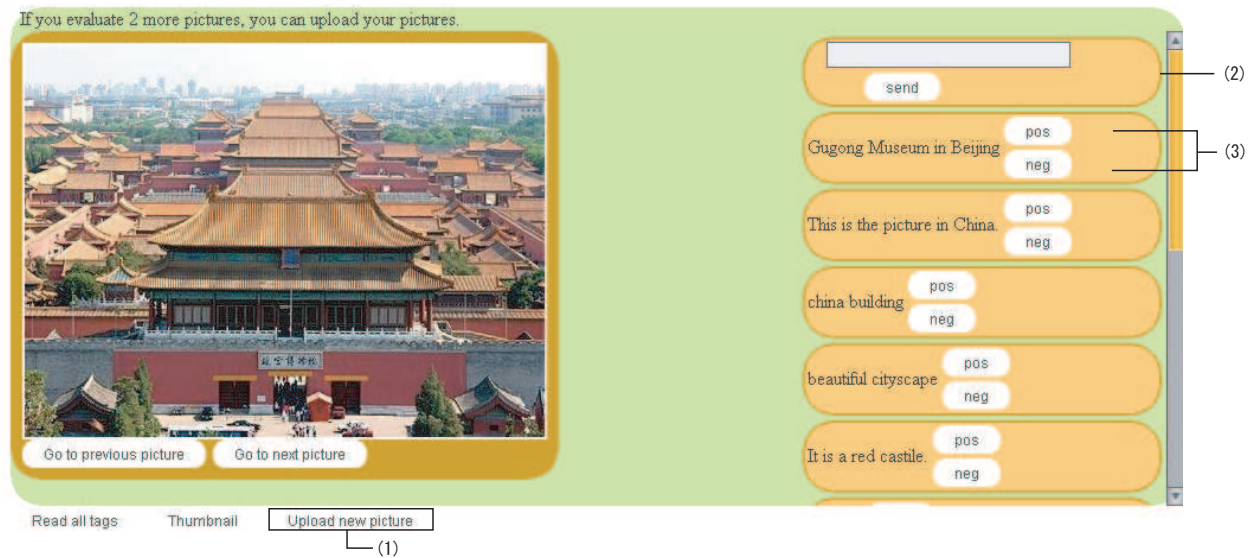


Figure 4.1: Screen shot

4.3 Experiment Overview

Designing an effective GA for tagging system requires evaluating implemented operators; mutation and selection. The designed GA was intent to increase three objects; for improving the qualities of tags, (1) the average number of fitness points on the tags; for creating better tag than user-typed one, (2) the maximum number of fitness point on the tag; for extending the initial population size, (3) the number of user-typed tags. This experiment was done from April 7, 2008 to April 13, 2008.

The procedure is as follows.

1. Send e-mails to the same participants as the preliminary experiment
2. Each participant writes tags
3. Each participant evaluates tags
4. Each participant uploads pictures

Repeat 2 to 4 for a week. Mutation and selection are performed at the step 3. The step 2 and 3 was done simultaneously.

4.4 Experimental Results

This section describes the experimental results. Figure 3 depicts the number of tags assigned to all the pictures for each day. A preliminary experiment was done in order to see how tags are evaluated by users. Table 0.13 shows the average, standard deviation, minimum and maximum of the point score, the number of accesses, and the fitness of tags for all the upload pictures. Figure 5 and Figure 4 show the results with elitist scheme and without elitist scheme, respectively. The result indicates elitist scheme successfully preserved the tags assigned by users. Tags in the preliminary experiment are used in both cases.

Table 0.13 shows statistics of tags from both the preliminary experiment and the experiment. As shown in the table, the number of tags per user is successfully increased.

Table 0.13 shows the average, standard deviation, minimum and maximum of fitness points, the number of accesses, and the fitness of all user typed tags from the experiment.

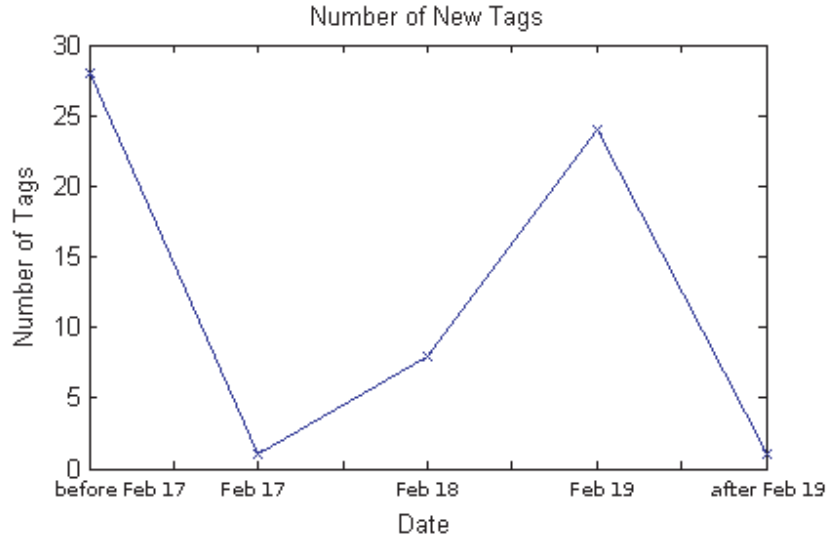


Figure 4.2: Amount of new tags

Table 4.1: Preliminal experiment results

	Average	Std Deviation	Minimum	Maximum
points	3.03	12.17	-64	44
# of access	64.90	60.52	10	222
points/#of access	0.045	0.15	-0.74	0.51

Table 4.2: Unique Users

	Prototype	Experiment
Number of Users	8	15
Number of User Wrote Tags	32	81
Number of New Tags	32	103
Avg Tags/User	4.0	5.4

Table 4.3: Experiment Results of User Typed Tags

	Average	Std Deviation	Minimum	Maximum
points	1.22	2.88	-2	13
# of access	28.30	35.23	2	203
points/#of access	0.042	0.099	-0.047	0.50

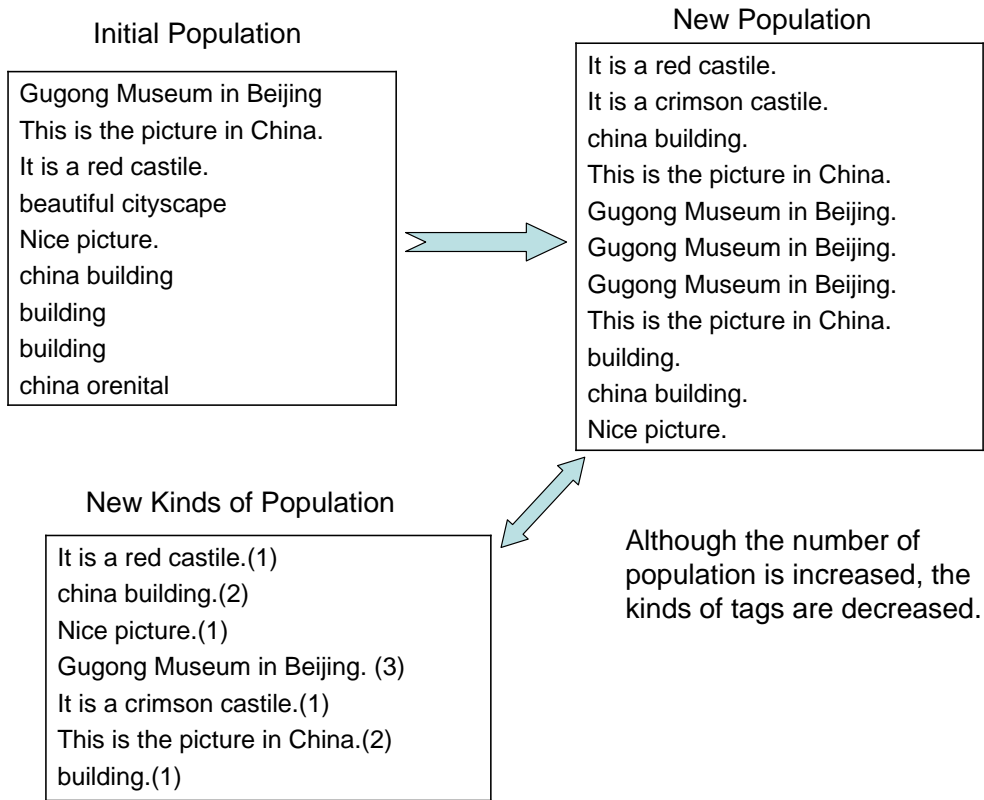


Figure 4.3: Selection without Elitist Scheme

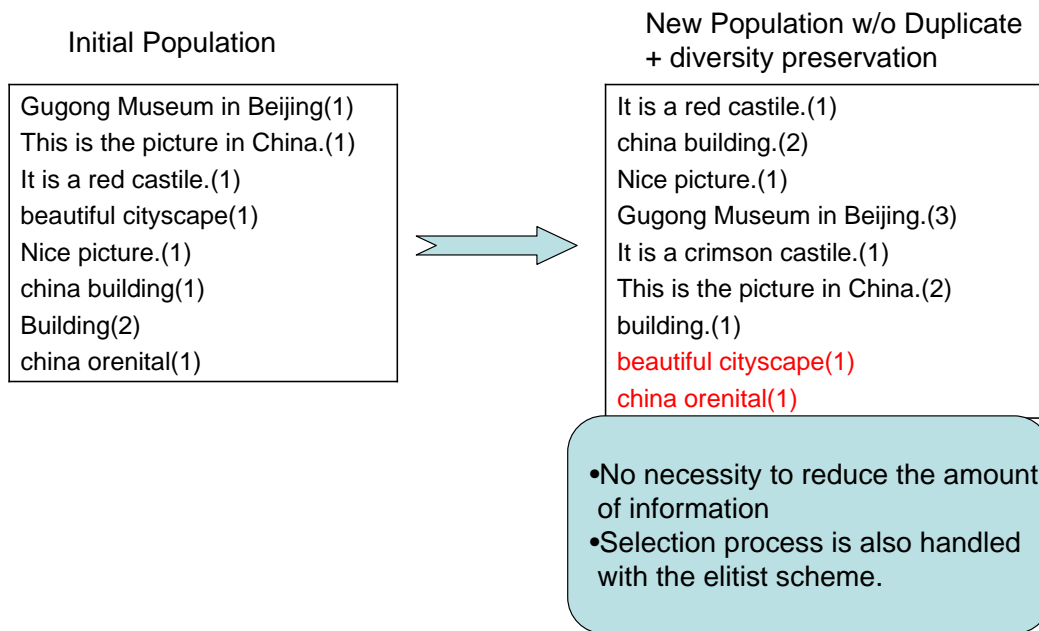


Figure 4.4: Selection with Elitist Scheme

Compared with the prototype, the average points is decreased. This is because the system operates mutation and selection after users send the evaluations to the system. While users cannot do anything for 30 seconds. However, the points/number of access is nearly the same. Although users have to wait for 30 seconds, this does not affect the quality of evaluation. In contrast to the prototype, the average points divide by the number of access is increased by 31

Table 0.13 shows the average, standard deviation, minimum and maximum of points, the number of accesses, and the fitness of all user typed tags and mutated tags from the experiment.

Table 4.4: Experiment Results of all New Tags (Includes Mutated Sentences)

	Average	Std Deviation	Minimum	Maximum
points	2.08	3.29	-2	13
# of access	36.22	43.09	2	267
points/#of access	0.059	0.102	-0.047	0.50

Overall, the proposed tagging system improves the fitness of tags. However, this does not achieve to create a “better sentence than the user typed” since both maximum points/# of access is 0.50. This is because the relatively small mutated tags (22 sentences), and the rapid decrease of accesses after several days as same as the prototype shows.

4.5 Future Work for Developing Recombination Operator

This chapter discussed the proposed collaborative tag builder system. In order to use GA for building better tags, initial population, fitness, and two operators were described. The GA was implemented in the system. The implementation of recombination operator is left for future work. In order to have recombination operations in the system, we need to have a larger number of users and more frequent access to the system. Since recombination increases the varieties of tags, fitness evaluation becomes more difficult.

Chapter 5

Conclusion

This thesis proposed a methodology toward creating a collaborative tag builder. The current trends of Web 2.0 for rapid growth interaction of users and introducing computational models which enables semantic web are explained. Thereafter, an idea of a tag builder is introduced for integrating user interactions and computational models.

Chapter 2 presented a brief introduction to genetic algorithms, and explained how GAs help integrate user interactions and computational models. User interaction is adapted to outsource fitness evaluation in order to enhance the integrity of users with helps of computational models. Chapter 3 demonstrated conceptual clustering. This is the step for handling computational models in the proposed system since the proposed methodology enables decomposing annotations into conceptual clusters. Chapter 4 demonstrated the tagging method which is based on the interactions of users. The method consists of three parts: (1) identifying object to fit; (2) implementing mutation and selection operators, and (3) operating them for improving the suitability of tags.

This thesis demonstrated the effectiveness of the proposed system that can integrate user-generated knowledge and computer-generated knowledge while enhancing the interactions between users and systems. However, boosting user innovations with the help of computational models did not work well. Implementing a recombination operator into the collaborative tag builder was proposed as a solution to this problem.

More work is needed beyond the simple pilot experiments presented here, but the study suggests that techniques borrowed from genetic algorithms and evolutionary computation may be useful in a variety of Web 2.0 applications. It is hoped that these studies inspire and help guide future work.

References

- Braz, R. d. S., Girju, R., Punyakanok, V., Roth, D., & Sammons, M. (2006). An inference model for semantic entailment in natural language. *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, Revised Selected Papers, 3944*, 261–286.
- Bush, V. (1945). As we may think. *The Atlantic Monthly*, July.
- Caldwell, C., & Johnston, V. (1991). Tracking a criminal suspect through "face-space" with a genetic algorithm. In *ICGA* (pp. 416–421).
- Fernandez, C. (1998). *Integration analysis of product architecture to support effective team co-location*. Master's thesis, Massachusetts Institute of Technology, Boston, MA.
- Goldberg, D. (1993). Making genetic algorithm fly: A lesson from the Wright brothers.
- Goldberg, D. (2002). *Design of innovation: Lessons from and for competent genetic algorithms*. Boston, MA: Kluwer Academic Publishers.
- Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.
- Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6, 333–362. (Also IlliGAL Report No. 91010).
- Goldberg, D.E., Welge, M., & Llorá, X. (2003, June). *DISCUS: Distributed Innovation and Scalable Collaboration In Uncertain Settings* (IlliGAL Report No. 2003017). Urbana, IL: University of Illinois at Urbana-Champaign.
- Harik, G., Lobo, F., & Sastry, K. (1999). *Linkage learning via probabilistic modeling in the ecga* (IlliGAL Report No. 99010). Urbana, IL: University of Illinois at Urbana-Champaign.
- Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.
- Kosorukoff, A. (2001, March). *Human based genetic algorithm* (IlliGAL Report No. 2001004). Urbana, IL: University of Illinois at Urbana-Champaign.
- Loper, E., & Bird, S. (2002). Nltk: The natural language toolkit. In *ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics* (pp. 62–69). Somerset, NJ.
- Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge, UK: Cambridge University Press.
- Miller, G., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. (1990). Introduction to wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4), 235–244.
- Mitchell, T. (1997). *Machine learning*. New York: The McGraw-Hill Companies, Inc.
- Moldovan, D., Harabagiu, S., Girju, R., Morarescu, P., Lacatusu, F., Novischi, A., Badulescu, A., & Bolohan, O. (2002). Lec tools for question answering. In *10th Text REtrieval Conference (TREC-2001)* (pp. 355–361). Gaithersburg, MD.

- Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 275–281). Melbourne, Australia.
- Porter, M. (1980). An algorithm for suffix stripping. *Program*, 14(3), 130–137.
- Rissinen, J. (1978). Modeling by shortest data description. *Automatica*, 14, 465–471.
- Saruwatari, S., Llorà, X., Goldberg, D., Yasui, N., Sastry, K., & Hiroshi, T. (2008). Speeding online synthesis via enforced selecto-recombination.
- Sastry, K. (2007). *Genetic algorithms and genetic programming for multiscale modeling: Applications in materials science and chemistry and advances in scalability*. Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL.
- Sastry, K., & Goldberg, D. E. (2004). Designing competent mutation operators via probabilistic model building of neighborhoods. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2, 114–125. Also IlliGAL Report No. 2004006.
- Sharman, D., Yassine, A., & Carlile, P. (2002, Sept.). Characterizing modular architectures. ASME 14th International Conference, DTM-34024.
- Song, F., & Croft, W. B. (1999). A general language model for information retrieval (poster abstract). In *Proceedings of the eighth international conference on Information and knowledge management* (pp. 279–280). Kansas City, Missouri.
- Steward, D. V. (1981). The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, 28, 77–74.
- Takagi, H. (2001, September). Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9), 1275–1296.
- Ueda, T., Llorà, X., Goldberg, D., Yasui, N., & Sastry, K. (2007). Discovering building blocks for human based genetic algorithms. In *Intelligent Engineering Systems Through Artificial Neural Networks* (pp. 183–188). New York, NY: ASME PRESS.
- Walker, M. A., Rambow, O., & Rogati, M. (2001). Spot: a trainable sentence planner. In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001* (pp. 1–8). Morristown, NJ, USA: Association for Computational Linguistics.
- Weiss, S. M., Indurkha, N., Zhang, T., & Damerou, F. J. (2005). *Text mining: Predictive methods for analyzing unstructured information*. New York: Springer.
- Yassine, A., Falkenburg, D. R., & Chelst, K. (1999). Engineering design management: An informatoin structure approach. *International Journal of Production Research*, 37(13), 2957–2975.
- Yu, T., Yassine, A., & Goldberg, A. (2005, March). *An Information Theoretic Method for Developing Modular Architectures Using Genetic Algorithms* (IlliGAL Report No. 2005014). Urbana, IL: University of Illinois at Urbana-Champaign.