

Automating Keyphrase Extraction with Multi-Objective Genetic Algorithms

Jia-Long Wu
University of California at Berkeley
Berkeley, CA 94720
jialong@me.berkeley.edu

Alice M. Agogino
University of California at Berkeley
Berkeley, CA 94720
aagogino@me.berkeley.edu

Abstract

Keyphrases have been used extensively in IR systems to facilitate information exchange, organize information and assist information retrieval. Automation of keyphrase generation is essential for the timely creation of keyphrases for large repositories in new domains where previous thesauri do not exist or for metacollections in which keyphrases that are meaningful across disparate collections are needed. In this paper we propose an automated keyphrase extraction algorithm using a non-dominated sorting multi-objective genetic algorithm. The "clumping" property of keyphrases is used to judge the appropriateness of a phrase and is quantified by a condensation clustering measure proposed by Bookstein. The objective is to find the smallest phrase set that has the best precision, as measured by average condensation clustering. Keyphrases were retrieved from a collection of design conference papers and the results were presented to domain experts for evaluation. Ninety percent of the generated phrases were deemed appropriate for use in a thesaurus for engineering design.

1. Introduction

With the exponential rate of growth of digital information, improving the efficiency of document retrieval from digital document repositories is a key research issue in information science. Previous research has shown that using keyphrases to assist the information retrieval processes can increase the retrieval efficiency of the system. Fagan [1] and Mitra [2] both show that phrase-based indexing helps to increase the precision of the overall retrieval, especially at lower relevance ratings, even though it does not significantly improve the result at higher relevance ratings where phrases tend to over-emphasize a particular aspect of the query.

Anick [3] developed a relevance feedback system *Paraphrase* based on the observation that "lexical

dispersion" of a term – the number of different noun compounds containing that term appearing in a given document set – predicts the likelihood of the term being a topical term. He claims that noun compounds, i.e., keyphrases, could provide more discriminating power compared to individual terms in the compound. Phrases are extracted from the initial result set and presented to the user for relevance feedback. Based on similar ideas, Jones [4] developed *Phrasier*, a document retrieval system that ranked the similarity between documents, where the query is also treated as a document, by the degree of keyphrases overlapping in the documents. Results from human evaluation of *Phrasier* indicate that the phrase-based retrieval system is as effective as a full-text retrieval system while requiring less storage space for indexing. Buckland et al. [5] use noun phrases in the titles, authors and/or abstracts and metadata vocabularies such as classifications to build what they call an Entry Vocabulary Module (EVM). EVM associates the noun phrases and the metadata vocabularies with the likelihood ratio statistic and serves as a dictionary for translating user queries in ordinary language (English, in this case) to corresponding metadata vocabularies that are unfamiliar to the user. With EVM, a user can easily search over repositories indexed using different vocabularies and also documents in different languages without any prior knowledge about the language used.

Since utilizing phrases is beneficial in many aspects in information retrieval, finding a set of key phrases for a document repository that is suitable for both indexing and document retrieving is essential to improving efficiency of retrieval systems. For moderate size document repositories one can manually scan through all the documents and select the most useful phrases for the system. However, for a large document repository, e.g., a web portal like Yahoo or a design document database at Boeing, performing this task manually is immense and costly. Being able to extract key phrases automatically from digital document repositories becomes a necessity in these cases.

Keyphrases also play an important role in exchanging and retrieving information with multiple repositories. The

Unified Medical Language System (UMLS) project [6] was initiated to overcome the difficulties in communicating efficiently between medical information systems using different vocabularies. A crucial component of UMLS is a thesaurus consisting of key concepts in the medical domain. This thesaurus serves as the metadata for vocabularies from different sources, and is referred to as metathesaurus in UMLS. With this metathesaurus in place, vocabularies can be mapped to the keyphrases in the metathesaurus which, in turn, can be translated into vocabularies in another IR system. UMLS built the thesaurus by continuously having domain experts searching and evaluating the keyphrases from the academic literature, and medical reports and other documents in the medical domain. An automated keyphrase extraction tool can reduce the efforts required by experts tremendously. By relieving the experts from the tedious task of scanning through documents and extracting possible keyphrases, they will be able to put more efforts into evaluating the candidate keyphrases generated by the extraction tool and updating the metathesaurus more efficiently.

In this paper, we exploit the clumping property of the keyphrases proposed by Bookstein [10] and form the extraction task as an optimization problem. The objective is to find an optimal set of keyphrases from a pool of candidate phrases that is most “clumped” in the document sets. The Multi-Objective Genetic Algorithm (MOGA) approach was chosen to tackle this problem for several reasons. First, the problem can be easily modeled within a genetic algorithm framework, compared to gradient-based nonlinear programming or simulated annealing. Second, MOGAs provide a natural framework for explicitly considering multiple objectives without the need to formulate a single composite multiobjective function. Third, the population of solutions can provide trade-off information between objectives and also a basis for sensitivity analysis of individual phrases on the optimal solutions.

2. Related research in automated keyphrase extraction

Several key phrase extraction techniques have been proposed and implemented successfully in different contexts [7-9]. Some exploit the syntactic nature in languages and others approach the problem by studying the statistical significance of the key phrases in the document sets they reside. Bookstein and Picard have both found that keyphrases are often clustered together and appear in a similar context with other key phrases more often than not. Bookstein [10] proposes two measures to evaluate the candidate keyphrases. The *condensation clustering* measure reflects the degree of “clumping” of candidate phrases in the context and the

linear clustering measure recognizes the patterns of textual units within which the candidate phrases reside. Results from Picard’s paper [11] suggest that content-bearing key phrases display higher first and second order similarity between one another and are less similar to the noise phrases.

As part of the InfoFinder Agent, Krulwich and Burkey [7] extract significant topic phrases from documents deemed relevant by the users to serve as surrogates of user interests. Based on the extracted phrases, the agent induces a decision tree and uses the decision tree to form a Boolean query. Documents that are related to the “learned” user interests can then be retrieved with this Boolean query. Visual patterns that might convey the author’s intention to capture the reader’s attention – such as all capital letters – are used to decide semantically significant phrases. Other heuristic metrics include phrases in different fonts, in quotes or parenthesis, or words including punctuation or numerical characters. This approach requires very few computation resources but may result in low recall and precision in keyphrase retrieval. However, the learning stage that follows the phrase extraction can correct some of the problems and sift out unwanted phrases.

Nevill-Manning et al. [12] use an algorithm called SEQUITUR to build hierarchical structures of phrases from a sequence of words. The algorithm replaces any phrase in the sequence which appears more than once with a grammatical rule and does it recursively. The integrity of the structure is maintained by following two constraints: digram uniqueness and rule utility. The algorithm makes sure no two digrams are the same and also every rule is used more than once. The hierarchical structure of phrases is then displayed in the PHIND interface in a way that allows users to traverse the structure to refine query terms. SEQUITUR can process large chunks of text in a short period of time and the hierarchy of phrases developed provides a framework to assist users in browsing large text collections. However, as the goal of SEQUITUR is to improve browsing by providing the user with a “good idea of the subject matter”, the algorithm is not designed to develop an optimal set of noun phrases useful for a comprehensive thesaurus. In addition, phrase boundary conflicts may overlook important phrases and memory requirements for processing large text collections are reported to be high.

In *GenEx*, Turney [8] proposes a mechanism composed of two parts: Extractor and Gantor. The Extractor takes in documents and produces keyphrases based on 12 parameters. These parameters include a desired number of output phrases, thresholds for the position of the first occurrence of the phrase, length of the stemmed phrase, etc. With the training set, Gantor uses a steady-state genetic algorithm to find an optimal set of parameters that allows the Extractor to extract the best set

of keyphrases, that is the one that has the most matches to the known keyphrase set in the training document set. Once the training is done, Genitor is no longer needed for the trained domain and keyphrases can be extracted with Extractor alone using the trained parameters. Turney also tried another approach to extract keyphrases using the C4.5 decision tree induction algorithm with 9 syntactic features, concluding that *GenEx* performed better than the C4.5 algorithm. The extracted keyphrases were evaluated via a web interface by human experts and 80% of them were deemed good keyphrases.

Frank et al. [9] propose using another machine learning algorithm based on a “naïve” or simplified Bayes’ rule to extract keyphrases. In the *Kea* system they developed, two attributes are used to classify phrases: (1) the term frequency-inverse document frequency (*tf-idf*) score of the phrase and (2) the distance into the document of the phrase’s first occurrence. Assuming independence of the attributes, a Bayes’ model is constructed with the attributes from the training document set. Candidate phrases generated by examining the phrase boundaries in the target document are then fed into the model and the probability of the phrase being a keyphrase is calculated. Ranking of the phrases based on the probability is reported to the user and a certain number of keyphrases are chosen based on the user’s preference. Frank et al. shows that *Kea* requires less computational time and achieves the same or better results than *GenEx*. Both *Kea* and *GenEx* require a separate training document set with keyphrases already carefully assigned in order to function properly. For some domains, such training sets might be difficult to obtain.

3. Finding keyphrases with a multi-objective genetic algorithm

GenEx and *Kea* both work well for generating keyphrases for individual documents. However, in many applications, there is no appropriate document set that can be used to train the algorithm. Furthermore, both algorithms focus on finding keyphrases inside individual documents, rather than for the whole corpus. It is possible to aggregate the keyphrases assigned to each document and use the resulting keyphrase union set to represent the corpus. However, aggregation might result in unnecessary redundancy of similar phrases and inclusion of specialized words relevant to a single document, but not useful for a representative set of keyphrases for the corpus. In addition, these algorithms generate a large number of keyphrases and use a “cut-off” number for each document. Where to place the “cut-off” to optimize the keyphrases for the corpus is not clear.

Instead we propose to formulate the task of generating keyphrases for the whole corpus as a global optimization problem. The objective is to find the optimal set of

keyphrases that can best represent the corpus. As the “optimal” condition itself is usually multi-faceted and may have competing facets, we hypothesized that multi-objective genetic algorithms (MOGAs) would be a promising approach. Instead of choosing the “cut-off” on the number of keyphrases heuristically, MOGAs can provide information on the trade-offs between the number of phrases and the metrics that are used to evaluate the phrases. The set of Pareto solutions with different trade-offs can be useful in that some keyphrase applications may prefer a larger phrase set with acceptable precision whereas other applications might benefit from a smaller, precise phrase set. In the following sections we describe our step-by-step procedures for applying the MOGA algorithm to keyphrase extraction.

3.1. Candidate phrase extraction

Before applying a genetic algorithm to select keyphrases, candidate phrases have to be extracted from the target document collection. Textual units are taken out of the documents and broken down into sentences. We used the Apple Pie Parser developed at the New York University [13] to extract all noun phrases from the sentences. These phrases are compared to a list of stop words (phrases) and all the phrases that are on the list are eliminated from the set. All candidate phrases are converted to lower case characters. Term frequencies, inverse document frequencies, and clumping metrics of the phrases are also calculated and stored for later use.

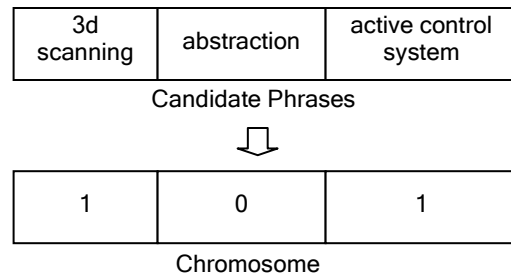


Figure 1. Representation of Chromosomes

3.2. Modeling keyphrase extraction within genetic algorithms

Each chromosome in the population represents a candidate selection of key phrases. The length of the chromosome is the number of candidate key phrases extracted from the pre-process stage. Each gene in the chromosome represents a candidate phrase and can take on a value of 1 or 0. If the candidate phrase is selected, the corresponding gene will have an allele of 1, 0 otherwise. Figure 1 is an example of part of the chromosome and its relationship with the candidate

phrases. For example, “3d scanning” and “active control system” are selected as content-bearing phrases. Their corresponding genes are encoded with allele of 1 while the gene representing “abstraction” has the value of 0.

3.3. Fitness function

Bookstein [10] found that content-bearing terms have a tendency to be clustered or clumped and developed a measure of the condensation of the term over textual units. If a term is clustered, they hypothesize that fewer textual units would have the term in them. Thus the number of textual units that contain a term should be less than the total number of occurrences of the term for content-bearing terms. This statistical significance can be exploited and used as part of our objective function to find an optimal set of keyphrases. Bookstein proposed a condensation measure based on the differences in probability of finding a phrase that is content-bearing in a textual unit and the probability of finding one that is not content-bearing. The expected number of textual units containing the phrase can be found as

$$E = D[1 - (1 - \frac{1}{D})^T] \quad (1)$$

where D is the number of total textual units in the repository, and T is the total occurrence of the phrase. One measure associated with condensation clustering of the phrase is then defined as

$$M_c = N/E \quad (2)$$

where N is number of textual units that actually contains the phrase. We call this the measure of *dispersion* of the phrase over the textual units in the document. If the phrase is randomly distributed over the textual units, the dispersion would be unity. If the phrase is clustered, then its dispersion over the textual units will be less than one; the smaller the dispersion measure the larger the clustering tendency.

In our multi-objective formulation we chose to minimize the average dispersion of the selected phrases in order to optimize the clumping property, and thus the precision potential, of our extracted keyphrases. As we mentioned before, another benefit of using a MOGA is to provide the trade-off information between the number of phrases and the evaluation metric. In our test example, the second objective in our algorithm is the number of phrases selected.

3.4. Multi-objective genetic algorithm (MOGA)

Deb et al. [14] developed an elitist non-dominating sorting genetic algorithm called the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) in 2000. The offspring population is first created by using the parent population through a crowded tournament selection, where the better individuals in the parent population, the

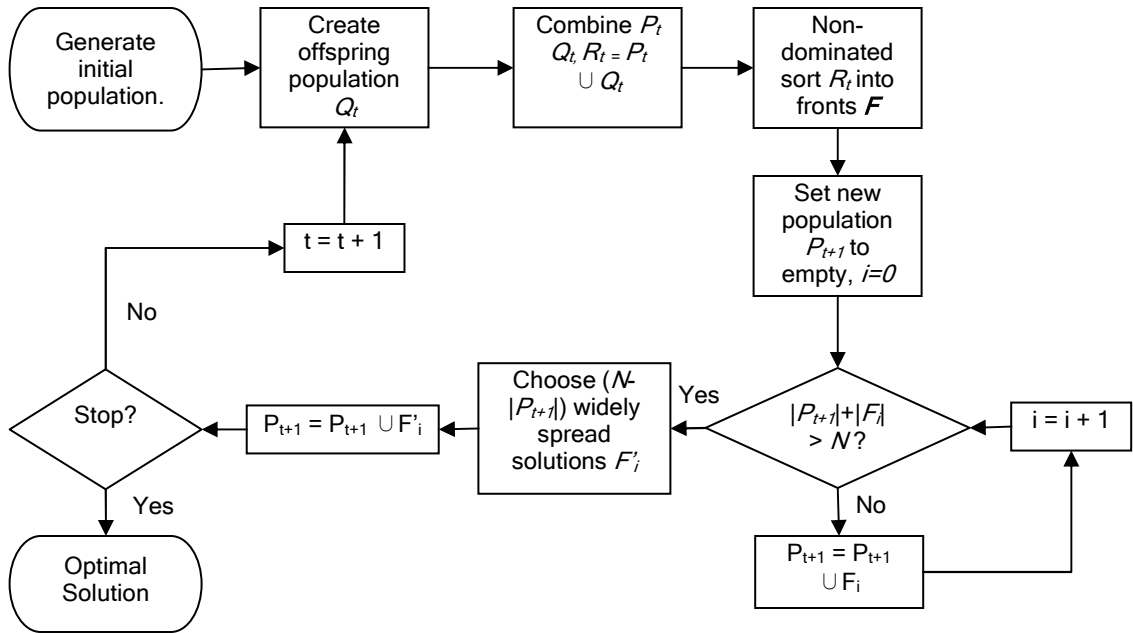


Figure 2. Schematics of the NSGA-II procedure

“elites”, are selected in a way that also strives to maintain the diversity in the population. Selected individuals will then go through crossover and mutation operations to form an offspring population. Both offspring and parent populations are then combined and sorted into non-dominated fronts. Among individuals in each front, there is no one single best solution. Each one of them performs better in some objectives than other individuals, but worse in the remaining objectives. However, individuals in worse fronts are dominated by all individuals in the better fronts, meaning they are all sub-optimal solutions compared to the solutions in the better fronts. The next generation is then filled with the individuals from the sorted fronts starting from the best. If a front can only partially fill the next generation, crowded tournament selection is invoked again to ensure diversity. This strategy is called “niching”. Once the next generation population has been filled, the algorithm loops back to creating an offspring population from this new parent population. Figure 2 details the procedure of NSGA-II.

4. Experimental results

One of the applications that inspired this project is the generation of a thesaurus for engineering design. Although the number of “born digital” engineering design documents has been steadily increasing, efficient storage and retrieval systems for them are still lacking due to difficulty in indexing the documents properly. Furthermore, differing terminologies used in design documents written by diverse groups of engineers can be confounding [15]. A thesaurus for engineering design that is similar to the Medical Subject Heading (MeSH) would be of great help in developing a user interface that can assist non-domain expert users to retrieve domain specific documents successfully and also serve as the backbone of the organization of the document repositories. The information flow between repositories or retrieval from multiple sources can benefit from such thesauri as well.

With this goal in mind, we selected all of the papers from the American Society of Mechanical Engineers (ASME) Design Theory & Methodology 2001 (DTM ‘01) Conference as our data set. There were a total of 34 papers at DTM ‘01 and we extracted 4,921 distinct noun phrases after performing the aforementioned pre-process procedures. Because of the length of the chromosome, we implemented a two-point crossover operation with the crossover probability of 0.6. For the mutation operation we used a Gaussian distribution and a 0.01 mutation probability. The initial population size was 100 and maintained for the following generations. Although NSGAI does not guarantee convergence because of the niching technique implemented, we did find the algorithm consistently converged at around 5000 generations for our optimization problem. On a dual Xeon 1.8GHz

workstation running linux, the algorithm took about 5 hours.

In figure 3 we plotted all the solutions in the final population by average dispersion versus number of phrases selected. The concave curve is the Pareto front and all the solutions on the curve are considered to be equally optimal. If the application imposes a limitation on the number of phrases due to storage or presentation restrictions, e.g., limited space on the printed version of academic journals or proceedings, a solution on the upper-left end of the Pareto curve should be chosen with a certain degree of sacrifice in the overall quality of the set of phrases. If a better set of keyphrases is desired and the number of phrases is not of great concern – as in the case of generating indexing terms for a digital document repository – one might select solutions on the lower-right hand side of the Pareto curve – solutions with low dispersion (and thus high condensation) but a high number of phrases. Also, a sensitivity analysis could be performed based on the Pareto curve in order to evaluate the solutions based on individual costs imposed on each of the objectives.

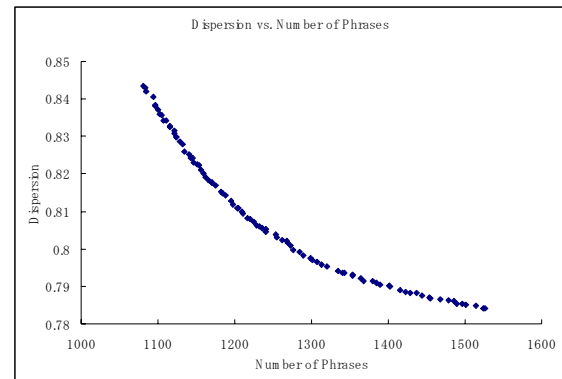


Figure 3. Pareto plot: dispersion vs. number of phrases

Another benefit of the MOGA algorithm is that we can gain additional information by observing the makeup of the population of Pareto optimal solutions. In the population of the final generation, the number of selected phrases ranged from 1,000 to 1,500 out of the original 5,000 candidate phrases. Aggregating the selected phrases from all the solutions, we found that 2,156 unique phrases were selected in the final population. 528 phrases out of 2,156 appeared in all solutions, which make them the most likely keyphrases. Figure 4 shows the distribution of phrase occurrences in the final population.

An interesting trend in figure 4 is that occurrences of phrases concentrate in both ends of the spectrum. About half of the selected phrases occur in the majority of the Pareto optimal solutions and the rest only appear in less than 10% of the final population. We hypothesize that the phrases with higher occurrences are the ones that

constitute the core of the solution while the low-occurrence ones serve as the tuner for the trade-off between the number of phrases and the quality of the phrase set. More detailed experiments are needed to test this hypothesis and ascertain the cause of this phenomenon.

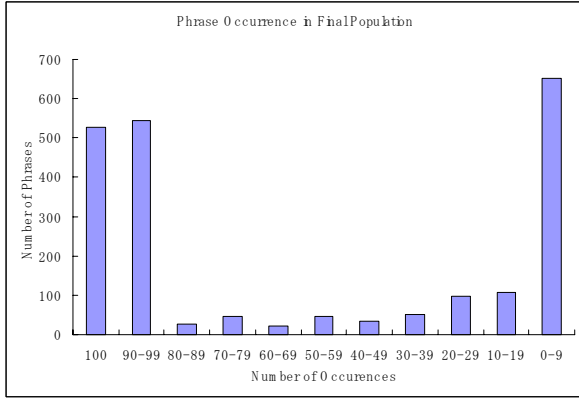


Figure 4. Distribution of phrase occurrences in final population

5. Human Evaluation of Generated Keyphrases

We first eliminated phrases that could be automatically pruned (proper nouns, singular duplicates and stemming redundancy). 777 out of 2,156 phrases were removed from the set. The remaining 1,379 phrases included 385 *core phrases* that appeared in all solutions. As our goal was to evaluate the adequacy of the generated phrases as keyphrases for a corpus of research papers in engineering design theory and methods, we concluded that an evaluation by a mix of domain experts in this field would be appropriate.

Two different web interfaces were designed for the evaluation: one site contained all 1,379 phrases and the other only the 385 core phrases. Evaluators were asked to check the phrases that they thought were not appropriate keyphrases for engineering design. We assigned about equal number of evaluators to each phrase list and recorded the phrases they deemed as unacceptable keyphrases. A total of six evaluators responded to our invitation. All evaluators are professors, researchers or graduate students who were familiar with the field of engineering design. The evaluation results are shown in table 1.

We investigated whether the *core phrases* – those that appear in all solutions (figure 4) – fair better than the rest of the phrases. Results were separated into relevant core phrases and relevant non-core phrases. The overall quality of the generated phrases seems to be quite high. On average less than 10% of the phrases were deemed not related to engineering design. Core phrases do appear to

be more relevant percentage-wise, but not significantly. Notice the large deviation for both set of phrases. We asked several evaluators to reflect on their selection criteria after submission. As we asked them if the phrases were relevant to general engineering design and not to a specific topic or a specific document, some of the evaluators found it rather difficult to build a mental standard for judging. They thought some phrases might be useful in certain contexts – for example the word “refrigerator” in “refrigeration design – but might not be useful for a corpus with a broader design scope. Some evaluators tried to be more inclusive and only eliminated those phrases that did not make sense at all while others were more exclusive and only recognized the phrases that were unique to engineering design as the relevant ones. This explains the wide dispersion in the number of relevant phrases.

Table 1. Human evaluation of relevancy of extracted keyphrases

	Relevant Core Phrases (out of 385 candidates)	Relevant Non-Core Phrases (out of 994 candidates)
Average from 6 Experts	363.5	905.5
Percentage Relevant	94.42%	91.10%
Standard Deviation	13.08	74.77

We also performed a comparison of our automatically generated keyphrases to the author assigned keyphrases for our test corpus. It was interesting to note that only half of the papers had author-assigned keywords, motivating the need for automatic generation. We also found that the author-assigned phrases were sometimes broader conceptual terms than the actual phrases used in their articles. Each paper with keyphrases had on average 4 keyphrases assigned and about 20% of the author-assigned keyphrases did not appear in any of the papers, including the paper with the assignment. Approximately 80% of the author-assigned keyphrases that were used in the papers were included in the list of generated keyphrases.

6. Conclusions and Future Research

Keyphrases have long been recognized as good surrogates for documents and concepts and are used extensively in browsing and query reformulation in information retrieval systems to improve user satisfaction in retrieved results. In order to allow for multiple objectives over a corpus of documents, we framed the

automated keyphrase extraction problem as a global multi-objective optimization problem. We exploited the “clumping” property of keyphrases as proposed by Bookstein [10] and implemented a multi-objective genetic algorithm to find the optimal set of keyphrases that displayed the lowest dispersion (highest condensation) level and minimized the number of phrases. Using our framework, additional preferences could be added to the optimization model if needed. Trade-off information between different objective functions can be gained from the final generation.

A human evaluation procedure was carried out to assess the quality of extracted phrases. Our results report that over 90% of the phrases were acceptable keyphrases for engineering design. In addition, 80% of the author-assigned keyphrases used in the documents showed up in the generated lists as well. The preliminary results indicate that our proposed MOGA algorithm can extract a reasonable good keyphrase set just by processing a collection of documents in a particular domain without any prior training or domain-specific knowledge. We believe such an approach could greatly reduce the effort of developing domain-specific thesauri and updating established thesauri more efficiently. We envision that the generated phrases as a whole can be an efficient indexing tool as well as a tool for reformulating user queries through query expansion. A subset of the generated phrases that are pertinent to a certain topic can also be used to assist authors or editors assigning keywords to academic papers on that topic.

By examining the distribution of occurrences of keyphrases in the final population, we found that phrases tend either to be present in almost all solutions or exist in very few solutions. We hypothesize that phrases that have fewer occurrences are the ones that can be used to tune the optimals along the Pareto curve. A detailed sensitivity analysis will be needed to validate this hypothesis.

A more extensive evaluation with a larger number of raters and a six-scale rating system is currently underway to better assess the range in quality of the generated phrases. Results to date show that fewer than 5% of the generated terms are considered inappropriate, consistent with the results obtained from the binary rating evaluation summarized in Table 1.

We are also adding to our Design Theory corpus ASME DTM papers for multiple years, along with papers from the International Conference on Engineering Design (ICED). To validate the usability of keyphrases in a real application, we are implementing an interface for retrieving DTM conference papers and suggesting keywords to the authors based on the automatically generated keyphrases.

Comparison with base-line IR systems that use full-text search will also be conducted to evaluate the added value of the generated keyphrases for initial and refined

queries. An evaluation will also be conducted on a large established dataset – such as one of the TREC (Text Retrieval Conference) test collections – to evaluate the scalability and portability of our algorithm.

In addition to developing a thesaurus for the Design Theory community we hope to apply this technique to develop a metathesaurus across engineering disciplines in the area of engineering education for the NSF-funded National Science, Technology, Engineering & Mathematics Digital Library (NSDL). The evaluators, in this case, would be engineering faculty and students in higher education.

7. Acknowledgements

The authors would like to thank Dr. Andy Dong in the Computational Design unit in the School of Architecture at the University of Sydney for his suggestion and guidance in the use of genetic algorithms for thesaurus generation. We are grateful to Dr. William H. Wood in Mechanical Engineering at University of Maryland at Baltimore County for his vision in recognizing the need for metathesaurus generation for the NEEDS/SMETE digital library project and for his collaboration on strategies for its evaluation. This research was funded, in part, by the National Science Foundation under Grants #DUE-0121743 and #DUE-0127580 for the National STEM Education Digital Library (NSDL) program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

8. References

- [1] J.L. Fagan, “The Effectiveness of a Nonsyntactic Approach to Automatic Phrase Indexing for Document Retrieval,” *Journal of the American Society for Information Science*, 40(2), 1989, pp. 115-132.
- [2] M. Mitra, C. Buckley, A. Singhal and C. Cardie, “An Analysis of Statistical and Syntactic Phrases,” *Proceedings of RIAO97, Computer-Assisted Information Searching on the Internet*, Montreal, Canada, 1997, pp. 200-214.
- [3] P.G. Anick, “The Paraphrase Search Assistant: Terminological Feedback for Iterative Information Seeking,” *Proceeding of SIGIR '99*, 1999, pp. 153-161.
- [4] S. Jones, “Phrasier: a System for Interactive Document Retrieval Using Keyphrases,” *Proceedings of SIGIR99*, 1999, pp. 160-167.
- [5] M. Buckland et al., “Mapping Entry Vocabulary to Unfamiliar Metadata Vocabularies,” *D-Lib Magazine*, <http://www.dlib.org/dlib/january99/buckland/01buckland.html>, 1999.

- [6] National Library of Medicine, Unified Medical Language System, sixth experimental edition., Bethesda, MD, 1995.
- [7] B., Krulwich, & C., Burkey, "The Infofinder Agent-Learning User Interests through Heuristic Phrase Extraction," IEEE Intelligent Systems & Their Applications, 12(5), 1997, pp. 22-27.
- [8] P.D. Turney, "Learning Algorithms for Keyphrase Extraction," Information Retrieval, 2, 303-336, 2000.
- [9] I. Witten, E. Frank, G. W. Paynter, "Domain-Specific Keyphrase Extraction," Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 1999, pp. 668-673.
- [10] A. Bookstein, S.T. Klein, T. Raita, "Clumping Properties of Content-Bearing Words," Journal of the American Society for Information Science, 49(2), 1998, pp. 102-114.
- [11] J. Picard, "Finding content-bearing terms using term similarity," Proceedings of Ninth Conference of the European Chapter of the Association for Computational Linguistics, Bergen, Norway, 1999.
- [12] C. Nevill-Manning, I. Witten and G. Paynter, "Browsing in Digital Libraries: a Phrase-Based Approach," Proceedings of 1997 International Conference on Digital Libraries, 1997, pp. 230-236.
- [13] S. Sekine and R. Grishman, "A Corpus-based Probabilistic Grammar with only Two Non-terminals," *Fourth International Workshop on Parsing Technologies*, Prague, 1995.
- [14] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan, "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II," Proceedings of the Parallel Problem Solving from Nature VI (PPSN-VI), 2000, pp. 849-858.
- [15] A. Messac and W. Chen, "The Engineering Design Discipline: Is its Confounding Lexicon Hindering its Evolution?" Proceedings of DETC '98, DTM-5658, 1998.