A General Optimization Framework for Multi-Document Summarization Using Genetic Algorithms and Swarm Intelligence

Maxime Peyrard and **Judith Eckle-Kohler**
Research Training Group AIPHES and UKP Lab
Computer Science Department, Technische Universität Darmstadt
www.aiphes.tu-darmstadt.de, www.ukp.tu-darmstadt.de

Abstract

Extracting summaries via integer linear programming and submodularity are popular and successful techniques in extractive multi-document summarization. However, many interesting optimization objectives are neither submodular nor factorizable into an integer linear program. We address this issue and present a general optimization framework where any function of input documents and a system summary can be plugged in. Our framework includes two kinds of summarizers – one based on genetic algorithms, the other using a swarm intelligence approach. In our experimental evaluation, we investigate the optimization of two information-theoretic summary evaluation metrics and find that our framework yields competitive results compared to several strong summarization baselines. Our comparative analysis of the genetic and swarm summarizers reveals interesting complementary properties.

1 Introduction

Extractive multi-document summarization (MDS) is often cast as a discrete optimization problem where the document collection is considered as a set of sentences and the task is to select an optimal subset of the sentences under a length constraint. Many successful approaches solve this problem using integer linear programming (ILP) or submodular function maximization. Both ILP and submodular function maximization require that the objective function to maximize has certain properties: it needs to be submodular or linearly factorizable (for ILP). The commonly used optimization objective that combines maximizing the coverage of relevant units while minimizing their redundancy has been shown to be submodular by Lin and Bilmes (2011). Lin and Bilmes (2011) also proved that the summary evaluation metric ROUGE itself is submodular, which has been leveraged by Sipos et al. (2012) to optimize ROUGE directly via submodular function maximization. Peyrard and Eckle-Kohler (2016) optimize an approximation of ROUGE via ILP. Using ROUGE as an optimization objective in MDS is reasonable, because ROUGE-1 and ROUGE-2 have been shown to correlate well with the results of human evaluation (e.g., Owczarzak et al. (2012)), and are therefore good proxies for system summary quality.

The results we present in this paper start from the observation that ROUGE is just one possible proxy for summary quality – there are other automatic metrics to evaluate system summaries, which also correlate well with human judgments (Louis and Nenkova, 2013). For example, the Jensen Shannon divergence (JS divergence), an information-theoretic measure not relying on human-written summaries, compares system summaries with source documents regarding their underlying probability distribution of n-grams (Lin et al., 2006). However, unlike ROUGE, JS divergence is neither submodular nor factorizable (see, e.g., Louis and Nenkova (2013)), and consequently can not be optimized via ILP or submodular function optimization. Similarly, we can imagine other interesting, arbitrarily complex summary evaluation metrics, which are neither submodular nor factorizable, but which we might want to optimize. For example, we might be interested in combining several metrics such that the combination correlates well with human evaluation scores.

In order to solve a discrete optimization problem (which is NP-hard) in the general case, where the optimization objective does not have specific properties, we have to rely on search algorithms. A particular promising class of search algorithms to tackle this problem are metaheuristics (Bianchi et al., 2009). In this paper, we consider two kinds of metaheuristics – genetic algorithms and a swarm intelligence approach called Artificial Bee Colony – and propose a general optimization framework for MDS where the function to be optimized is a parameter and can be exchanged.

Our contributions can be summarized as follows: we present an optimization framework for extractive MDS that is able to optimize an arbitrarily complex objective function of input documents and a summary, without making any assumptions on its properties. For our framework, we developed two summarizers, one based on genetic algorithms, the other based on swarm intelligence. In our experimental evaluation, we investigate the optimization of information-theoretic summary evaluation metrics and find that our framework yields competitive results compared to several strong baselines. We also comparatively analyze the behavior of our two summarizers, and observe that there are several system summaries with very high ROUGE score, which have almost no sentences in common.

2 Background

Some branches of research on optimization study the landscape of functions to maximize (also called fitness landscape). When the landscape has particular properties, methods can be derived to find optima efficiently. For example, ILP leverages the linear properties (Schrijver, 1986) of an objective function. Recently, submodular functions have been extensively studied and simple algorithms have been proved to yield nice solutions (Krause and Golovin, 2014; Schrijver, 2003).

However, we might not want to restrict ourselves to particular kinds of functions, because often, the landscape of the objective function does not have easy to exploit properties (Bianchi et al., 2009). To maximize such functions, the optimization field developed techniques that do not make any assumption about their landscape (Blum and Roli, 2003; Wright, 1932; Fogel et al., 1966).

To solve such complex optimization problems, search-based techniques are used (i.e., they do not find the exact solution like ILP), which employ various strategies to efficiently explore the search space. The majority of the successful search-based techniques are biologically inspired and take efficient and adaptive optimization strategies used in nature as a role model.

These algorithms can roughly be divided into two main categories according to their biological inspiration: **Genetic algorithms** which simulate the evolution process, and **swarm intelligence** algorithms which simulate the behavior of swarms and colonies of animals such as bees, ants or fireflies.

Genetic Algorithms In artificial intelligence, genetic algorithms (GA) use mechanisms inspired by biological evolution, such as reproduction, mutation and selection (Wright, 1932; Goldberg, 1989).

The candidate solutions of the optimization problem are represented as individuals in a population. The fitness function is the function to optimize and determines the quality of an individual. To apply a GA to an optimization problem, the parameters that define a candidate solution are considered as the genotype of the individual. The genotype is the footprint that uniquely identifies every possible solution.

Evolution of the population takes place after multiple iterations where biological operators are applied: Reproduction and mutation search the space of solutions by creating new candidate solutions, while the selection operator ensures that better candidate solutions survive to the next generation more often than worse ones.

Swarm Intelligence Swarm Intelligence (SI) refers to the collective behavior of decentralized, self-organized artificial systems. The agents in such systems follow very simple rules, and although there is no centralized control structure, local and random interactions between agents lead to the emergence of intelligent global behavior, unknown to the individual agents themselves (Beni and Wang, 1993; Bonabeau et al., 1999; Parsopoulos and Vrahatis, 2002).

While the population of the GAs consists of candidate solutions, the swarm population is made up of agents which search the solution space and interact locally with the environment. The candidate solutions are points in the space investigated by the agents. The agent interactions with the candidate space usually consist of the evaluation of a given area. To apply a SI algorithm to an optimization problem, one must define a space where candidate solutions live. A point in the space of candidate solutions is analogous to the vectors of genes (parameters, i.e., genotype) of the GA. Simple communication channels allow agents to exchange information about promising areas. Examples of such natural systems are ant colonies, fireflies glowing, fish schooling or bird flocking.

One successful model we decided to follow in this work is the model of honey bees searching for nectar in a field, also known as Artificial Bee Colony (ABC) (Karaboga and Basturk, 2007; Karaboga et al., 2014). In ABC, there are three groups of bees: employed bees, onlookers and scouts.

There is only one employed bee per food source (the number of employed bees in the colony is equal to the number of food sources investigated in parallel). Employed bees collect food from their food source and dance in this area after evaluating the quantity of food in the direct neighborhood. The dance indicates the amount of food in the area identified by the employed bee. When the food source is abandoned, the employed bee becomes a scout and starts to search for a new food source elsewhere. Onlookers watch the dances of employed bees and choose food sources which are especially promising. The overall behavior allows the swarm to find areas which contain a lot of nectar (or food). The employed bees and onlookers use the nectar function to evaluate the quantity of nectar at a given location (i.e., candidate solution) which is the equivalent to the fitness function in the GA.

We will study the differences and similarities of these algorithms when we adapt them to extractive summarization in the following section.

3 Optimization Framework

We propose a general framework that extracts a summary with high score for any metric that is a function of the source documents and the summary only. The metric to maximize is a parameter of our framework that can easily be changed. First, we present the metrics we considered in our framework, and then the optimization techniques we used.

3.1 Metrics

In our framework, we consider classical similarity metrics for comparing the source documents and the summary content.

As it was shown in previous work, good summaries are often characterized by a low divergence between the probability distributions of n-grams in the source documents and the summary (Haghighi and Vanderwende, 2009; Louis and Nenkova, 2013). One common metric for determining this divergence is the Kullback Leibler (KL) divergence. The KL divergence between two probability distributions P and Q is given by:

$$KL(P\|Q) = \sum_{g} p_P(w) log_2 \frac{p_P(w)}{p_Q(w)} \tag{1}$$

In the case of MDS, the two probability distributions P and Q are computed from the source documents and the summary.

Jensen Shannon (JS) divergence is a symmetric version of the KL divergence, incorporating the idea that the distance between two distributions cannot be very different from the average of distances from their mean distribution. It is given by:

$$JS(P\|Q) = \frac{1}{2}(KL(P\|A) + KL(Q\|A)) \tag{2}$$

where $A = \frac{P+Q}{2}$ is the mean distribution of P and Q.

The metrics are defined on the basis of n-grams, and are not restricted to words only. In practice, we use both the unigram and bigram version of these metrics. It is important to point out that these metrics only serve as example metrics – any other metric function of the input documents and the summary can be plugged into the framework.

3.2 Biologically-Inspired Optimization Techniques

In this section, we will describe how to adapt GA and SI to the problem of extractive summarization.

Genetic Summarizer In order to produce a Genetic Summarizer, we use a simple analogy to the problem of extractive summarization.

- **Population** The individuals of the population are the candidate solutions which are valid extractive summaries. Valid means that the summary meets the length constraint. The size of the population is a hyper-parameter of the algorithm.

- **Genome and Genotype** The genome is the set of sentences in the source documents of the topic. It corresponds to the building blocks of each possible individual. Then, the genotype of a summary is simply a binary vector indicating which sentences it contains.

- **Fitness Function** The fitness function which evaluates the individuals (i.e., summaries) is the function we wish to maximize (or minimize). For example, it might be one of the metrics we described above. The population is scored and sorted according to the fitness function, a threshold indicates which summaries will survive to the next generation. The survival rate is another hyper-parameter.

- **Mutation** The mutation of a summary is done by randomly removing one of its sentences and adding a new one that does not violate the length constraint. Mutation affects individuals of a population randomly, and the mutation rate is a hyper-parameter.

- **Reproduction** The reproduction is done by randomly selecting parents among the survivors of the previous generation. Then, the union set of the sentences of the parents is considered. The child is a random valid summary extracted from these sentences. Similar as for the mutation, we define a reproduction rate which controls the number of children in each generation.

- **Initial Population** The initial population is simply created by randomly building valid summaries. We observe a convergence speed-up if we help the algorithm by including good summaries in the initial population (e.g., summaries produced by baseline algorithms).

We note that in nature, the fertilized egg cell undergoes a process known as embryogenesis before becoming a mature embryo. This is believed to make the genetic search more robust by reducing the probability of fatal mutation. At each step, we only consider valid summaries, which is the direct analogy to embryogenesis.

Swarm Summarizer

- **Food Location** The locations in the field are the candidate solutions which are the valid extractive summaries. The number of food locations is a hyper-parameter equivalent to the size of the population in the Genetic Summarizer.

- **Location Coordinates** The summaries are points in the space searched by the bees. The coordinates are given by the binary vector indicating which sentences the summary contains. The coordinates of a food location is the equivalent to the genotype of an individual.

- **Nectar Function** The nectar function which evaluates the food locations (i.e., summaries) is the function to maximize. It corresponds to the fitness function.

- **Employed Bees Local Search** At each iteration, employed bees evaluate the direct neighborhood of their assigned food location. To move to a neighbor, a sentence is randomly removed from the current summary (i.e., food location) and a new sentence that does not violate the length constraint is added. This is the analogy to a mutation.

- **Employed Bees Dance and Onlooker Bees** When employed bees have evaluated the summaries, all the summaries are scored and sorted. Onlooker bees observe this distribution of scores and randomly choose one location to join and help with the neighbor search. This random choice is based on the following probability:

$$P_i = \frac{score_i}{\sum_k score_k} \tag{3}$$

  As a result, onlookers choose high scoring locations (i.e., summaries) more often.

- **Scouting Bees** An employed bee stays in place if the neighbor it evaluates is not better than its current location. After several iterations at the same place, it abandons it and becomes a scouting bee. To move to another place, the scouting bee selects a random valid summary. This is equivalent to generating an individual from the initial population in the Genetic Summarizer. The number of iterations before becoming a scouting bee is the second hyper-parameter of the Swarm Summarizer.

Genetic vs. Swarm In the Genetic Summarizer, the reproduction produces significant changes in the summaries studied, because, on average, half of the genotype of the child is different from its parents. But at the same time, it stays in a reasonable distance range from its parents because it keeps half of the genotype from each parent (on average). In this sense, we say that the Genetic Summarizer has efficient mid-range search capabilities. The local search is much reduced because it is done via mutations happening randomly and rarely in the population. The long-range search is done via insertion of random individuals into the population whenever the population becomes too small. A new completely random individual is quite likely to have a low fitness score and to die in the next generation with few opportunities to reproduce or mutate.

The Swarm Summarizer has complementary strengths and weaknesses. The employed bees perform intensive local search around a specific location and the onlooker bees help them around the locations of interest. For long-range search, the scout bees regularly look for new locations and investigate each new area for at least t rounds (where t is the hyper-parameter controlling the number of retry before becoming a scout bee). However, in the Swarm Summarizer, the mid-range search is limited compared to the reproduction mechanism, because it is achieved only by either successfully applying several local movements, or by randomly scouting the mid-range areas, both of which are unlikely.

Even if we did not conduct an extensive hyper-parameters optimization, we observe that the Swarm Summarizer has much less hyper-parameters, which would make it simpler to optimize.

4 Experiments

4.1 Summarizer Performance

Baselines: We compare our algorithms to several classical baselines:

TF*IDF weighting This simple heuristic was introduced by Luhn (1958). Each sentence receives a score from the TF*IDF of its terms and the best sentences are greedily extracted until the length constraint is met.

LexRank (Erkan and Radev, 2004) is a popular graph-based approach. A similarity graph $G(V, E)$ is constructed where V is the set of sentences and an edge e_{ij} is drawn between sentences v_i and v_j if and only if the cosine similarity between them is above a given threshold. Sentences are scored according to their PageRank score in G. We use the implementation available in the sumy package.[1]

ICSI (Gillick and Favre, 2009) is a recent system that has been identified as one of the state-of-the-art systems by Hong et al. (2014). It is a global linear optimization framework that extracts a summary by solving a maximum coverage problem considering the most frequent bigrams in the source documents. Boudin et al. (2015) released a Python implementation (ICSI sume) that we use in our experiments.

KL-div Greedy Haghighi and Vanderwende (2009) presented a greedy algorithm to minimize the KL-divergence of extracted summaries. The approach iteratively selects a sentence s_i to be included in the summary S, which is done by picking the sentence that minimizes the KL divergence between the

[1]https://github.com/miso-belica/sumy

	DUC-02		DUC-03		DUC-02		DUC-03	
	JS-1	JS-2	JS-1	JS-2	R-1	R-2	R-1	R-2
TF*IDF	0.4075	0.5522	0.4714	0.6144	0.4072	0.1201	0.3222	0.0660
LexRank	0.3618	0.5454	0.4208	0.6010	0.4311	0.1388	0.3574	0.0793
KL-Greedy	0.3805	0.5531	0.4413	0.6036	0.3945	0.1125	0.3231	0.0715
JS-Greedy	0.3683	0.5475	0.4386	0.5961	0.4299	0.1455	0.3312	0.0640
ICSI	0.3537	0.5107	0.4045	0.5657	0.4434	0.1556	0.3763	0.0947
JS-Gen	0.3196	0.5125	0.3776	0.5727	0.4397	0.1416	0.3728	0.0855
JS-Gen-2	**0.3026**	**0.4937**	0.4059	0.5476	**0.4545**	**0.1629**	**0.3787**	**0.0959**
JS-Swarm	0.3244	0.5326	**0.3751**	0.5651	0.4433	0.1352	0.3761	0.0879
JS-Swarm-2	0.3089	0.5004	0.4027	**0.5402**	0.4321	0.1507	0.3615	0.0893
KL-Gen	0.3325	0.5314	0.3810	0.5724	0.4470	0.1505	0.3658	0.0873
KL-Gen-2	0.3811	0.4952	0.4171	0.5556	0.3903	0.1369	0.3593	0.0893
KL-Swarm	0.3455	0.5311	0.3859	0.5604	0.4451	0.1499	0.3703	0.0876
KL-Swarm-2	0.3572	0.5140	0.3934	0.5543	0.4449	0.1574	0.3698	0.0897

Table 1: Performance of biologically inspired algorithms to minimize JS and KL divergence.

word distributions of the original input documents and $S \cup s_i$. We also implement **JS-div Greedy** which is the same greedy algorithm, but minimizes JS divergence instead.

Experimental Setup: In our experiments, we use two datasets from the Document Understanding Conference (DUC) (Over et al., 2007), namely the datasets from 2002 and 2003 (DUC-02 and DUC-03). We compare algorithms with the JS divergence metric for both the unigrams (JS-1) and the bigrams (JS-2) variants. The results are shown in Table 1. For completeness, we also report the ROUGE scores identified by Owczarzak et al. (2012) as strongly correlating with human evaluation methods: ROUGE-1 (R-1) and ROUGE-2 (R-2) recall with stemming and stopwords not removed. Our algorithms are denoted by F-A-(2), where F is the function minimized, A is the algorithm used (Genetic or Swarm), and 2 is present if the bigram version of the function is minimized.

We use a standard implementation of JS divergence with stemming to get the JS divergence scores. ROUGE scores are obtained with the ROUGE-1.5.5 toolkit.[2]

Results Analysis: We observe that Genetic and Swarm summarizers have similar performances, there is no statistically significant difference when evaluated with JS and ROUGE evaluation metrics.

Both summarizers outperform the greedy versions by a large margin in all experiments. This shows that the Genetic and Swarms summarizers are strong search algorithms for extractive summarization.

We compare the behavior of Swarm and Genetic in more detail in section 4.2.

Our summarizers perform on par with the state-of-the-art algorithm ICSI in terms of ROUGE (no statistically significant difference). However, the Genetic and Swarm summarizers perform significantly better for the JS divergence evaluation metric. This suggests that there are several different high scoring summaries in terms of ROUGE, and our algorithms find summaries different from ICSI. We investigate this observation in more detail in section 4.3. ROUGE is known to have issues with distinguishing good summaries, and here we observe that JS is a complementary metric allowing us to distinguish high-scoring summaries.

The biologically inspired algorithms can even outperform ICSI and the other baselines for the ROUGE metric, thus confirming the correlation between JS divergence and ROUGE.

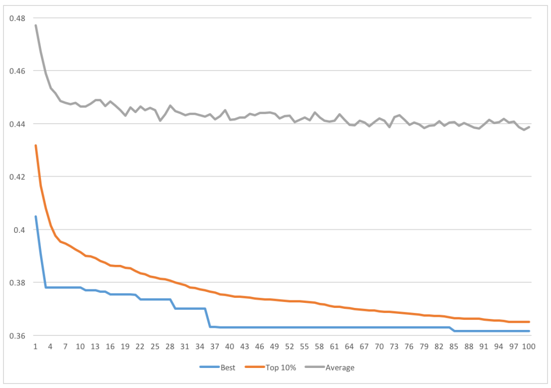
4.2 Convergence behavior

In this section, we study the behavior of our algorithms by investigating the convergence process of both the Genetic Summarizer and the Swarm Summarizer. For this experiment, we focus on the JS divergence minimization with the topic d30003t of DUC-2003.

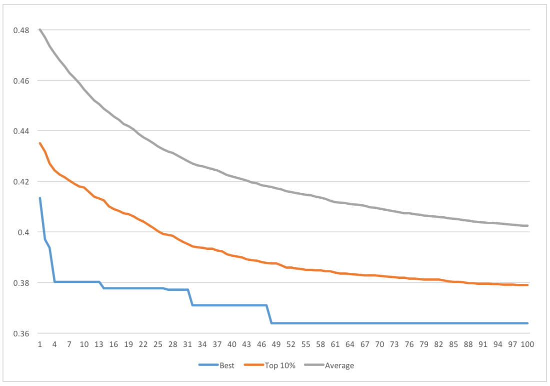
We plot in Figure 1a a graph displaying the evolution of three measures of the Genetic Summarizer across generations: the JS divergence of the best individual found, the average JS divergence of the top

[2]ROUGE-1.5.5 with the parameters: -n 2 -m -a -l 100 -x -c 95 -r 1000 -f A -p 0.5 -t 0. The length parameter becomes -l 200 for DUC-02.

10% of the population, and the average JS divergence of the population. The same analysis is reported for the Swarm Summarizer in Figure 1b.



(a) Genetic Summarizer convergence behavior (d30003t from DUC-2003).



(b) Swarm Summarizer convergence behavior (d30003t from DUC-2003).

Similarities: For both summarizers, the score of the best solution is quickly decreasing until some plateau is reached. Then, only punctual breakthroughs lead to significant improvements of the best solution. The top 10% of the population is regularly improving, following an exponentially decreasing curve, which makes it a good indicator that the algorithm has converged. Indeed, the best individual curve can plateau for several iterations and experience a large jump in one iteration. In Figure 1a, we observe a big breakthrough at generation 35 after a long period without improvement.

Differences: In the Genetic Summarizer, the average of the whole population quickly drops in the first few iterations and then stabilizes. In contrast, in the Swarm Summarizer, the curve of the overall average follows the curve of the top 10%.

The Genetic algorithm includes a lot of randomness at each generation. The majority of the population is randomly generated at each iteration (random restart and reproduction), and therefore the average of the population stays constant (there is a drop in the first iterations because the best members of the population are not random).

In the Swarm Summarizer, the randomness is controlled and the focus is put on local search. Employed bees and onlooker bees work on promising areas, and new random restarts (scouting bees) are introduced only when an area is not promising enough. By driving its workforce towards local search around promising areas, the Swarm Summarizer keeps working around better solutions.

For the Genetic summarizer, we also observe that the best elements of the population are closing the gap with the best individual, and the newly accumulated potential in the population allows for a new breakthrough via reproduction. The top 10% of the population are closer to the best solution than the top 10% of the Swarm summarizer.

With a good initialization (by introducing good summaries in the initial candidate solutions), the Swarm Summarizer converges much faster than the Genetic Summarizer. While the Genetic Summarizer keeps finding breakthroughs after 85 iterations, the Swarm Summarizer has converged already after 50 iterations.

In terms of runtime, both algorithms can find summaries close to the best ones after only several seconds. The Swarm Summarizer converges within about a minute for a given topic, while the Genetic Summarizer converges within 2-3 minutes on average [3].

4.3 Summaries Found

Our experiments in section 4.1 suggested that there are several different high scoring summaries. In this section, we study how different the summaries are from each other, and measure the diversity of the summaries discovered by our techniques.

[3]On a standard MacBook Pro with 16GB of RAM and i5 2.9GHz.

	JS-Gen	JS-Swarm
Jaccard to ICSI	0.0556	0.0610
Jaccard among top 10% (avg.)	0.4350	0.1786

Table 2: Diversity of summaries found by JS-Gen and JS-Swarm.

We know from section 4.1 that summaries produced by our algorithms and ICSI have many words and bigrams in common. However, since extractive MDS is a combinatorial problem, we can compare two summaries by comparing which sentences have been selected.

For this, we use the Jaccard distance between two summaries (each represented as set of sentences). It indicates the percentage of sentences two summaries have in common. We measure the Jaccard distance between the summaries of ICSI and the ones created by JS-Gen and JS-Swarm. To measure the diversity among the summaries found by our summarizers, we also measure the average Jaccard distance between two summaries belonging to the top 10% of solutions. The results are reported in Table 2. Surprisingly, we see that the summaries found by JS-Gen or JS-Swarm and those found by ICSI do use different sentences, as they have less than 10% of sentences in common – even though the JS-Gen and JS-Swarm summaries tend to use the same words and bigrams as ICSI. As they are not significantly different in terms of ROUGE scores, this indicates that there exist several extractive summaries with high ROUGE scores. In contrast, the JS divergence metric is able to distinguish between these summaries with high ROUGE scores.

We also observe that JS-Swarm works on more diverse summaries than JS-Gen, because the average Jaccard distance between summaries in the top 10% is much smaller compared to JS-Gen. JS-Swarm is capable of finding high scoring summaries in different areas and is therefore more likely to identify many of the different good summaries.

5 Related Work and Discussion

Previous applications of metaheuristics in the context of MDS used several different approaches to solve the MDS task. There are clustering-based approaches where metaheuristics are used to obtain a good sentence clustering (Aliguliyev, 2009; Song et al., 2011), and also ranking-based approaches where metaheuristics perform an optimization of an importance metric for sentence scoring (Litvak et al., 2010).

Most related to our framework is previous work where extractive MDS is also cast as a discrete optimization problem that is solved using metaheuristics. Several approaches applied genetic algorithms or variants to perform extractive MDS, but used different optimization objectives than us. Alguliev et al. (2013) employ differential evolution, an algorithm similar to GA where mutation is the main operation, and optimize a combination of content coverage and diversity. However, the runtime complexity of differential evolution is high and also depends on the runtime complexity of the fitness function used, which is high as well because their function includes a similarity comparison between sentences.

Nandhini and Balasundaram (2013) also use GA for extractive MDS, but consider the creation of assistive summaries rather than generic ones, and thus optimize a combination of readability, cohesion and topic-relatedness. He et al. (2006) use GA with an objective that combines the maximization of informativeness, the reduction of redundancy and also includes the length constraint. In contrast, our use of GA performs embryogenesis, i.e., non-valid individuals (summaries that do not have the required length) are not born.

While we considered KL and JS divergence as exemplary optimization objectives in our framework, any metric that is a function of the input documents and the summary can be used instead (e.g., other metrics from previous works), and also any combination of such metrics. Provided that human evaluation scores are available, we could even learn such a combination metric with the training objective that it correlates well with the human scores. This makes our optimization framework highly customizable to more specific summarization tasks, or genres other than newswire, where JS divergence might not be the best optimization objective (such as opinion and biographical texts as studied by Saggion et al. (2010)). To encourage the community to experiment with different metrics and summarization tasks, we provide the implementation of both the genetic and swarm summarizer at `https://github.com/UKPLab/`

`coling2016-genetic-swarm-MDS`.

6 Conclusion

We presented a general optimization framework for extractive MDS where any function of input documents and summary can be plugged in and used as an optimization objective. The algorithms we consider belong to the class of metaheuristics, and we developed an adaptation of genetic algorithms and of a swarm intelligence approach called Artificial Bee Colony to the task of extractive MDS. The Python implementation of the genetic and swarm summarizers is freely available[4] and can be extended with any other scoring function. As exemplary optimization objectives, we studied the summary evaluation metrics KL divergence and JS divergence. Our evaluation on the DUC-02 and DUC-03 datasets shows a competitive performance of our framework. In our detailed analysis we found interesting complementary properties of the genetic and swarm summarizers and of a strong baseline.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1, and via the German-Israeli Project Cooperation (DIP, grant No. GU 798/17-1).

References

Rasim M. Alguliev, Ramiz M. Aliguliyev, and Nijat R. Isazade. 2013. Multiple Documents Summarization Based on Evolutionary Optimization Algorithm. *Expert Systems with Applications*, 40(5):1675–1689.

Ramiz M. Aliguliyev. 2009. A New Sentence Similarity Measure and Sentence Based Extractive Technique for Automatic Text Summarization. *Expert Systems with Applications*, 36(4):7764–7772.

Gerardo Beni and Jing Wang. 1993. Swarm Intelligence in Cellular Robotic Systems. In *Robots and Biological Systems: Towards a New Bionics?*, pages 703–712, Berlin, Heidelberg. Springer.

Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. 2009. A Survey on Metaheuristics for Stochastic Combinatorial Optimization. *Natural Computing*, 8(2):239–287.

Christian Blum and Andrea Roli. 2003. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35(3):268–308.

Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. 1999. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA.

Florian Boudin, Hugo Mougard, and Benot Favre. 2015. Concept-based Summarization using Integer Linear Programming: From Concept Pruning to Multiple Optimal Solutions. In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1914–1918, Lisbon, Portugal.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality As Salience in Text Summarization. *Journal of Artificial Intelligence Research*, pages 457–479.

Lawrence J. Fogel, Alvin J. Owens, and Michael J. Walsh. 1966. Intelligent decision making through a simulation of evolution. *Behavioral Science*, 11(4):253–272.

Dan Gillick and Benoit Favre. 2009. A Scalable Global Model for Summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pages 10–18, Boulder, Colorado.

David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring Content Models for Multi-document Summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado.

[4]`https://github.com/UKPLab/coling2016-genetic-swarm-MDS`

Yan-xiang He, De-xi Liu, Dong-hong Ji, Hua Yang, and Chong Teng. 2006. MSBGA: A Multi-Document Summarization System Based on Genetic Algorithm. In *2006 International Conference on Machine Learning and Cybernetics*, pages 2659–2664.

Kai Hong, John Conroy, benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A Repository of State of the Art and Competitive Baseline Summaries for Generic News Summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1608–1616, Reykjavik, Iceland.

Dervis Karaboga and Bahriye Basturk. 2007. A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. *Journal of Global Optimization*, 39(3):459–471.

Dervis Karaboga, Beyza Gorkemli, Celal Ozturk, and Nurhan Karaboga. 2014. A Comprehensive Survey: Artificial Bee Colony (ABC) Algorithm and Applications. *Artificial Intelligence Review*, 42(1):21–57.

Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, pages 71–104.

Hui Lin and Jeff A. Bilmes. 2011. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 510–520, Portland, Oregon.

Chin-Yew Lin, Guihong Cao, Jianfeng Gao, and Jian-Yun Nie. 2006. An Information-Theoretic Approach to Automatic Evaluation of Summaries. In *Proceedings of the Human Language Technology Conference at NAACL*, pages 463–470, New York, NY, USA.

Marina Litvak, Mark Last, and Menahem Friedman. 2010. A New Approach to Improving Multilingual Summarization Using a Genetic Algorithm. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 927–936, Uppsala, Sweden.

Annie Louis and Ani Nenkova. 2013. Automatically Assessing Machine Summary Content Without a Gold Standard. *Computational Linguistics*, 39(2):267–300.

Hans Peter Luhn. 1958. The Automatic Creation of Literature Abstracts. *IBM Journal of Research Development*, 2:159–165.

K. Nandhini and S. R. Balasundaram. 2013. Use of Genetic Algorithm for Cohesive Summary Extraction to Assist Reading Difficulties. *Applied Computational Intelligence and Soft Computing*, 2013:8–16.

Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in Context. *Information Processing and Management*, 43(6):1506–1520.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An Assessment of the Accuracy of Automatic Evaluation in Summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9, Montreal, Canada.

Konstantinos E. Parsopoulos and Michael N. Vrahatis. 2002. Recent approaches to global optimization problems through Particle Swarm Optimization. *Natural Computing*, 1(2):235–306.

Maxime Peyrard and Judith Eckle-Kohler. 2016. Optimizing an Approximation of ROUGE - a Problem-Reduction Approach to Extractive Multi-Document Summarization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, volume 1: Long Papers, pages 1825–1836, Berlin, Germany.

Horacio Saggion, Juan-Manuel Torres Moreno, Iria da Cunha, Eric San Juan, and Patricia Velazquez-Morales. 2010. Multilingual Summarization Evaluation without Human Models. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2010)*, pages 1059–1067, Beijing, China.

Alexander Schrijver. 1986. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA.

Alexander Schrijver. 2003. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, New York, NY, USA.

Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Large-margin Learning of Submodular Summarization Models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 224–233, Avignon, France.

Wei Song, Lim Cheon Choi, Soon Cheol Park, and Xiao Feng Ding. 2011. Fuzzy Evolutionary Optimization Modeling and Its Applications to Unsupervised Categorization and Extractive Summarization. *Expert Systems with Applications*, 38(8):9112–9121.

Sewall Wright. 1932. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress of Genetics*, 1:356–66.