Automatic Annotation of Corpora for Text Summarisation: A Comparative Study

Constantin Orăsan

Research Group in Computational Linguistics,
School of Humanities, Languages and Social Sciences,
University of Wolverhampton, Stafford St.,
Wolverhampton, WV1 1SB, UK
C.Orasan@wlv.ac.uk
http://www.wlv.ac.uk/~in6093/

Abstract. This paper presents two methods which automatically produce annotated corpora for text summarisation on the basis of human produced abstracts. Both methods identify a set of sentences from the document which conveys the information in the human produced abstract best. The first method relies on a greedy algorithm, whilst the second one uses a genetic algorithm. The methods allow to specify the number of sentences to be annotated, which constitutes an advantage over the existing methods. Comparison between the two approaches investigated here revealed that the genetic algorithm is appropriate in cases where the number of sentences to be annotated is less than the number of sentences in an ideal gold standard with no length restrictions, whereas the greedy algorithm should be used in other cases.

1 Introduction

Annotated corpora are essential for most branches in computational linguistics, including automatic summarisation. Within computational linguistics, annotated corpora are normally considered a gold standard, and are used to train machine learning algorithms and evaluate the performance of automatic summarisation methods. In order to be used for these purposes, the annotation usually indicates the importance of each sentence. In this paper, the term gold standard is used only to refer to sets of sentences marked as important, and not to the whole annotated document. This approach was taken in order to facilitate the explanation to follow, and it does not prevent the methods presented in this paper being used to produce gold standards where the important sentences are annotated within the document.

The decision as to whether a sentence is important enough to be annotated can be taken either by humans or by programs. When humans are employed in the process, producing such corpora becomes time consuming and expensive. Methods which automatically build annotated corpora are cheap, but have some drawbacks. Section 2 presents brief details about the existing methods for producing such corpora for text summarisation.

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 670–681, 2005.

When corpora are used to evaluate automatic summarisation methods, the sentences extracted by an automatic method are compared with the ones marked by humans as important. In order to be able to have this comparison, it is usually necessary that the size of the automatic summary is the same as the size of the gold standard. Given that automatic summarisation systems can produce summaries which can have any compression rate, in order to evaluate them, gold standards should exist for all possible lengths. Because of the high costs of having people involved in the annotation process, it is not feasible to have the same text annotated for more than two or three different lengths. Automatic annotation methods can be applied in certain conditions, but the existing ones do not offer much control on the length of the gold standard.

This paper compares two automatic methods for producing annotated corpora for text summarisation. The advantage of these two methods is that they allow the automatic production of gold standards of predefined lengths. The structure of the paper is as follows: Section 2 briefly presents existing ways to produce annotated corpora for text summarisation. A greedy method inspired by [1] is explained in Section 3. Because the second method relies on a genetic algorithm, brief background information about genetic algorithms is presented in Section 4, followed by a description of the actual algorithm. Section 6 contains a comparative evaluation, and the paper finishes with discussion and conclusions.

2 Existing Methods for Building Annotated Corpora

Annotated corpora have been employed in automatic summarisation since the late 60s when Edmundson used one in the evaluation process. In order to produce the annotated corpus, Edmundson asked humans to identify the important sentences in each text from a collection of 200 scientific documents [2]. The important sentences were considered to be those which indicated *what* the subject area is, *why* the research is necessary, *how* the problem is solved, and *which* are the findings of the research. The annotators were asked to select 25% of the text, and to mark sentences in such a way that the selected sentences are coherent.

More recently, manual annotation was used to mark the important sentences in a corpus of newswire texts [3]. In this case, the annotators were required identify the main topic of a text, and to firstly mark 15% of the most important sentences, and then mark an additional 15% of the text. In order to maximise the coherence of the selected sentences, the annotators also marked sentences which are necessary for understanding the important sentences (e.g. sentences which introduce entities mentioned in the important sentences).

Given that identification of important sentences is very subjective and difficult, [4] and [5] took advantage of human produced abstracts, and asked annotators to align sentences from the document with sentences from the human produced abstracts. This set of sentences from the document is considered to convey the information from the abstract best, and therefore can be used as a gold standard. Kupiec et. al. [4] found that 79% of the sentences in the abstracts

could be perfectly matched with sentences from the full text, whereas Teufel and Moens [5] have observed that only 31.7% of the sentences from the abstracts have a perfect match. The percent of matching clauses is even lower in the experiment presented by Marcu [1]. One reason for these very dissimilar results could be the fact that the researchers worked with various types of documents, and did not use a common definition of a perfect match.

Annotated corpora can be automatically produced by using the observation that very often humans produce abstracts through *cut-and-paste* operations. As a result, it is possible to identify sentences from the document which are the source of the abstract. Marcu [1] combines a greedy algorithm with rules to recover this set of sentences. Marcu's method was reimplemented in this research and is explained in more detail in Section 3.

Jing and McKeown [6] treat the human produced abstract as a sequence of words which appears in the document, and reformulate the problem of alignment as a problem of finding the most likely position of the words from the abstract in the full document using a Hidden Markov Model.

3 A Greedy Method for Automatic Annotation of Important Sentences in Scientific Texts

Even though, it is difficult to identify direct matches between pairs of sentences from a document and its human produced abstract, it is generally agreed that it is possible to find several sentences from the document which were combined to produce a sentence in the abstract. Brute force approaches which try all the possible combinations are out of the question given that for a text with n sentences the number of possible extracts is $C_n^1 + C_n^2 + \ldots + C_n^n = 2^n - 1$.

In order to solve this problem, Marcu [1] proposes a greedy method. His method, instead of selecting sentences which are considered to be similar to those in the abstract, eliminates sentences which do not seem like the ones in the abstract. The underlining idea is that a sentence from a document does not resemble any sentence or part of sentence from the abstract if the similarity between the document and its abstract does not decrease when the sentence is removed from the document. This elimination process continues as long as such irrelevant sentences can be identified. When no more sentences can be eliminated, Marcu proposes a set of heuristics to further reduce the set of sentences left. After the rules are applied, the remaining sentences constitute the most similar extract to the human abstract, and can be used as a gold standard.

Algorithm 1 presents a modified version of Marcu's algorithm which was used here. The similarity between a set of sentences and the human produced summary is computed using the cosine similarity formula:

$$\cos(\boldsymbol{S_e}, \boldsymbol{S_h}) = \frac{\sum_{i=1}^{n} S_e(i) S_h(i)}{\sqrt{\sum_{i=1}^{n} S_e(i)} \sqrt{\sum_{i=1}^{n} S_h(i)}} \quad (1)$$

where S_e and S_h are the vectors built from the automatic extract and human abstract respectively, n is the number of distinct words in $S_e \cup S_h$, and $S_e(i)$ and

Data: Abstract = the human produced abstract, Source = $\{S_1, S_2, \ldots, S_n\}$, ExtractLen = the desired length or 0 if no limit is imposed
Result: Extract = a set of sentences from the source which has maximum similarity

```
1  Extract = Source;
2  Eliminate from Extract all the sentences with equations and references;
3  Eliminate from Extract all the sentences with less than 5 words;
4  while (Length(Extract) > ExtractLen) do
5  |  if (∃S ∈ Extract, Sim(Extract, Abstract) < Sim(Extract \ S, Abstract))
   |  then
6  |  |  Extract = Extract \ S;
7  |  end
8  |  else
9  |  |  break;
10 |  end
11 end
12 while (Length(Extract) > ExtractLen) do
13 |  Extract = Extract \ S, where Sim(Extract \ S, Abstract) > Sim(Extract \
   |  T), ∀T ∈ Extract, T ≠ S;
14 end
```

Algorithm 1. The elimination algorithm

$S_h(i)$ are the frequencies of word i in S_e and S_h respectively. In order to make the similarity value more accurate, stopwords were filtered, but no morphological transformation was used because it was desired that the gold standard uses very similar wording to the human produced abstract.

One difference between the original algorithm and the one used here is the fact that Marcu's heuristics employed to further reduce the set of sentences were not implemented. There are two reasons for not implementing them. Firstly, some of the heuristics require knowledge of the rhetorical structure of the source to be able to apply them. This information was not available, and could not be easily obtained. In addition, for some of the heuristics, the details were insufficient to know exactly how to implement them. Instead of the original heuristics, steps *2* and *3* were introduced to remove sentences with equations and references, and those which are very short.[1]

Another change which had to be made to the algorithm was to introduce a way to control the length of the gold standard. In the algorithm proposed by Marcu, there is no way to predict or restrict this length, and as a result, the method cannot be directly used to find a gold standard of a certain length.

In order to alleviate this problem, steps *12 – 14* were introduced in the algorithm. The role of these steps is to shorten the extract until the desired length is reached. This is achieved by identifying sentences whose elimination

[1] After several experiments, it was decided that a sentence which has less than 5 words, excluding punctuation, is not worth including in the extract.

Data: Fitness = a function which evaluates the quality of a chromosome,
PopSize = the size of the population
Result: Solution
1 P = The initial population with random chromosomes;
2 Evaluate the quality of the chromosomes in P;
3 **while** *(Termination condition not met)* **do**
4 Select chromosomes for recombination and put them in P_1;
5 Combine randomly selected chromosomes from P_1 and put the result in P_2;
6 Mutate randomly selected chromosomes from P_2 and put the result in P_3;
7 Produce the new population P from P_3 and the chromosomes not selected from P;
8 Evaluate the quality of the chromosomes in P;
9 **end**
10 Return Solution = the best chromosome selected according to the chosen strategy;

Algorithm 2. A typical genetic algorithm

cause the smallest decrease in the similarity between the selected sentences and the human produced abstract. In a number of situations, it happens that the length limit is reached in the main part of the algorithm (Steps *4 – 11*), and the algorithm returns the set of sentences which has been identified so far. Evaluation of the algorithm is presented in Section 6.

4 Brief Introduction to Genetic Algorithms

Genetic algorithms (GA) provide "a learning method motivated by an analogy to biological evolution" [7]. Introduced by Holland [8], genetic algorithms proved very effective in search and optimisation problems where the search space is complex. The way GAs work mimics reproduction and selection of natural populations to find the solution that maximises a function called *fitness function.*

There is no rigorous definition of a genetic algorithm but it is generally accepted that the common elements of GAs are: a population of chromosomes, a selection method based on a fitness function and genetic operators which evolve the population. A typical genetic algorithm is presented in Algorithm 2.

Chromosomes and Population: A *chromosome* encodes a possible solution to the problem to be solved. Each chromosome contains a fixed number of *genes*, which have binary, integer or real values, depending on the problem to be solved.[2] The length of a chromosome (i.e. the number of genes) and its meaning are also dependent on the problem. The population of chromosomes maintained by the

[2] There are algorithms where a chromosome contains a mixture of binary, integer or real values, or where the length of the chromosomes is variable. Because such algorithm are rarely used, and because this section is not supposed to offer a comprehensive overview of the existing GAs, they are not discussed here.

algorithm encodes different candidate solutions, and the number of chromosomes in the population varies according to the complexity of the search space.

Selection Method: The appropriateness of a chromosome is measured by a *fitness function* which is problem dependent. In addition to measuring the quality of a chromosome, the fitness function is also used to select which chromosomes will create a new population. The most common selection operator is called *fitness proportionate selection* because the probability of selecting a chromosome is proportionate to its fitness. Alternative selection methods are *elitist selection* where only the best chromosomes are used in the next generation, and *random selection* where the chromosomes are chosen on random basis.

Genetic Operators: After a set of chromosomes is selected, *genetic operators* are applied to produce a new population. The most common operators are *crossover* and *mutation*. The role of crossover is to take a pair of chromosomes, and produce two new offsprings by swapping or combining information from them. The mutation operator takes a single chromosome, and randomly changes the value of a randomly chosen gene. It has to be pointed out that each genetic operator has a certain probability to be applied, and as a result, not all the chromosomes are changed from one generation to the next.

Selection of the Solution: The goal of a genetic algorithm is usually to find a chromosome which maximises the fitness function. In order to achieve this, the population of chromosomes is evolved, and depending on the problem which is solved, the best chromosome found in all generations, or the best chromosome from the last generation is chosen as solution to the problem.

The number of iterations also depends on the problem to be solved. In some cases, the algorithm is expected to produce a predetermined number of generations, whereas in other cases, it stops when the fitness function reaches a certain threshold.

Genetic algorithms were successfully used in a wide range of fields including computational linguistics. In computational linguistics they were employed to improve the performance of anaphora resolution methods [9, 10], resolve anaphora resolution [11], study optimal vowel and tonal systems [12], build bilingual dictionaries [13], improve queries for information retrieval [14], and learn of syntactic rules [15]. For applications in other fields than computational linguistics a good overview can be found in [16].

5 A Genetic Algorithm for Automatic Annotation of Important Sentences in Scientific Texts

As shown in Section 6, the greedy algorithm presented in Section 3 preforms poorly when a length limit is imposed due to the iterative process which is employed to eliminate the sentences. Because of this, once a sentence is eliminated, it is impossible to undo the elimination. This section presents a genetic algorithm which determines the gold standard with a specific length.

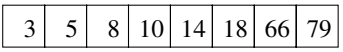

Fig. 1. A chromosome representing an extract which contains the sentences 3, 5, 8, 10, 14, 18, 66, 79 from the source

The genetic algorithm employed here does not work on a sentence by sentence basis. Instead, it encodes the whole set of sentences which might correspond to the gold standard in a chromosome. The length of the chromosome is the desired length of the gold standard, and each gene points to a sentence to be included in it. Figure 1 presents an example of a chromosome. With this encoding, caution needs to be taken whenever a new chromosome is produced so the values of the genes are distinct (i.e. the summary does not contain a sentence more than once). If a duplication is found in a chromosome, then the value of the gene which contains the duplication is incremented by one.

An alternative to this encoding is binary encoding, where the length of a chromosome is the number of sentences in the text. In this encoding, each gene corresponds to a sentence, and a value of 1 for the gene indicates that the sentence is to be included in the gold standard, whereas a 0 value designate a sentence not to be included in the gold standard. This encoding was not used here because the number of sentences in the documents is quite large, which means that the resulting chromosomes would have been too long, and in such cases the convergence of the genetic algorithm is not certain. The second difficulty that needed to be tackled with such encoding is how to produce gold standards of a certain length (i.e. to have an exact number of genes with value 1 in the chromosomes). Such encoding would have required an additional operator which checks the number of genes with value 1, but the decision on which genes should be changed to achieve the desired number of 1s seemed very arbitrary. In light of these problems, it was decided not to use binary encoding.

The fitness function employed by the genetic algorithm is cosine similarity between the set of sentences represented by a chromosome, and the human abstract. Given that the GA is trying to maximise the value of the fitness function, the set of sentences determined by the best chromosome is the most similar to the human abstract, and therefore can be used as a gold standard.

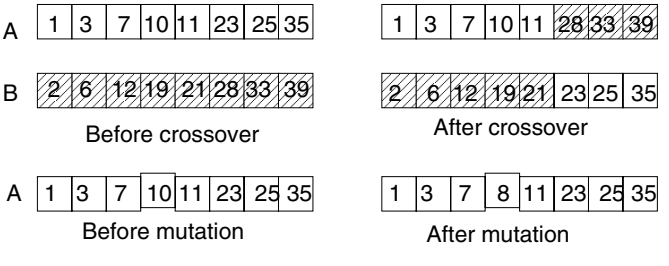

Fig. 2. The genetic operators

The genetic operators used for this particular problem are fitness proportionate selection, single point crossover and point mutation. Figure 2 shows how crossover and mutation operators work. After the crossover and mutation operators are applied, duplicate genes need to be identified. As a result, it is possible to say that two mutation operators are applied to the population.

6 Evaluation

The evaluation was performed using a corpus consisting of 65 files from the Journal of Artificial Intelligence Research (JAIR) with over 600,000 words. These texts were chosen because they contain human produced abstracts, and therefore they can be used by the methods described in this paper. From each document 2%, 3%, 5%, 6%, and 10% gold standards were produced. Using the greedy method, gold standards without a length limit were also generated. Table 1 presents the average similarity scores and their standard deviation obtained for different sets of sentences. The first column corresponds to the unrestricted gold standard, and can be determined only with the greedy algorithm. For the genetic algorithm, two different settings were tried in order to assess how the number of chromosomes influences the algorithm's convergence.

In order to have a better view of the methods' performance a baseline was also implemented. This baseline takes the first and the last sentence of each paragraph, starting with the first one, until the desired length is reached. This baseline was selected because it was noticed that important sentences in scientific articles tend to occur in these positions. A random baseline was considered too naive and easy to defeat to be employed here. Table 1 reveals that both methods perform better than the baseline.

Table 1. The average similarities between the *ideal* extract and the human produced abstracts

	Unrestricted	2%	3%	5%	6%	10%
Baseline						
Average similarity	-	0.260	0.327	0.419	0.440	0.479
Standard deviation	-	0.137	0.159	0.163	0.153	0.131
Greedy algorithm						
Similarity average	0.818	0.392	0.521	0.687	0.732	0.788
Similarity standard deviation	0.082	0.196	0.193	0.167	0.141	0.079
Genetic algorithm with a population of 500 chromosomes						
Average	-	0.720	0.734	0.738	0.738	0.733
Standard deviation	-	0.104	0.094	0.087	0.085	0.087
Genetic algorithm with a population of 2000 chromosomes						
Average	-	0.725	0.743	0.752	0.748	0.746
Standard deviation	-	0.103	0.092	0.088	0.084	0.084
The average lengths in sentences of the gold standards						
No. sentences	30.18	9.81	14.84	25.15	30.25	50.84

To assess whether there is a link between the length of the gold standards and the performance of different algorithms, the average lengths of the gold standards were computed. These values are presented in the last row of Table 1.

Investigation of the results obtained by the greedy algorithm reveals that the similarity scores obtained for the 2%, 3% and 5% gold standards are very poor in comparison with the rest of the results. As can be seen in the last row of Table 1, these gold standards are shorter than the ones with unrestricted lengths. For this reason, Steps *12 – 14* of Algorithm 1 are needed to reduce their length. As a result, a significant proportion of information is lost which suggests that Steps *12 – 14* are not the right way to control the length of the gold standard.

The solution of the genetic algorithm was chosen to be the best chromosome after 200 generations. Such a large number of generations was necessary because the search space is very large. Experiments with algorithms which iterated for 300 generation revealed that the improvement which can be obtained is negligible. Given the large search space, the number of chromosomes in the population was also large. In order to find out how this number influences the convergence of the algorithm, two experiments were run. In the first one, the population contained 500 chromosomes, and in the second one this number was increased to 2000.

As can be seen in Table 1, the results obtained with the genetic algorithm are much more homogenous than the ones of the greedy algorithm. As expected, the results for the genetic algorithm with 500 chromosomes are lower than the ones obtained with 2000. This can be explained by the fact that the latter can explore the search space more efficiently, and therefore can identify better solutions. In addition, it was noticed that the algorithm with 2000 chromosomes converges more rapidly to the solution.

Comparison between the results of the two methods reveals that the genetic algorithm is a much better option for determining the 2%, 3%, 5% and 6% gold standards, and only for 10% extracts it performs worse. The conclusion which can be drawn is that the genetic algorithm is a good way to determine the gold standard when its length is shorter than the length of an unrestricted gold standard, whereas the greedy algorithm is appropriate for gold standards which are longer than the unrestricted gold standard. The low standard deviation obtained for the genetic algorithm suggests that its results are more consistent across different texts. Figure 3 presents the sentences selected as important by the greedy algorithm and by the genetic algorithm. The human produced abstract is also displayed there.

7 Discussion and Conclusions

This paper has presented two automatic methods for building annotated corpora for text summarisation using the human produced abstracts which accompanies documents. The first method uses a greedy approach to eliminate those sentences which do not resemble the information present in the abstract, whilst the second one relies on a genetic algorithm to achieve the same goal. Because the set of sentences determined by the two methods contain similar information to the

human abstract, these sentences can be marked as important in the text, in this way obtaining a gold standard.

Given that the methods proposed in this paper can be applied only where there is a human produced abstract, one may believe that the usefulness of these methods is limited. This is not the case because the gold standards identified by the methods can be used to evaluate automatic extraction methods using precision and recall. Moreover, the gold standard can be used by machine learning algorithms to learn when to extract a sentence. None of these tasks can be done only on the basis of human abstract.

The results of the evaluation revealed that each method is appropriate for a different purpose. The greedy method is suitable in the cases where a gold standard with no length limit imposed has to be determined, and for the gold standards which have the length longer than the unrestricted one. The performance of the greedy algorithm degrades quickly with the decrease in the length of the gold standard, in all the cases where gold standards shorter than the unrestricted one need to be produced, the genetic algorithm should be used. In addition, low standard deviation noticed in the results indicates that the algorithm has a consistent performance across different documents.

As expected, the sets of sentences identified by the methods cannot match those marked by humans. This is normal given the simplicity of the process employed. However, the methods proposed in this paper can be particularly useful for long documents where it is not feasible to ask humans to mark the important sentences. Moreover, for scientific documents, it is necessary to have annotators who understand the topic of the text. In many cases this makes the annotation process more expensive because domain experts have to be employed.

One of the most common criticisms of genetic algorithms is their slow speed, especially in experiments where the population size or the number of generations are large. It is true that the genetic algorithm employed here is quite slow, but even with a population of 2000 chromosomes, it performs faster than the greedy algorithm. The explanation for this can be found in the iterative behaviour of the greedy algorithm.

An alternative use for the methods presented in this paper is to determine the upper limits of extraction methods when run on a document, provided that the evaluation method uses similarity to assess the quality of an extract. Because both methods try to maximise the similarity score between a set of sentences extracted from the text, and the human produced abstract, the maximum for the similarity score indicates the upper limit of any extraction method. It should be pointed out that given the nature of the two methods, none of them actually guarantees that the absolute maximum is reached. For both of them, it is possible that there is another set of sentences which obtains a higher similarity score, but the determined value will be very close to the absolute maximum.

The genetic algorithm presented in this paper can be used to determine not only one, but all the sets of sentences which have a maximum the similarity with the human abstract. This can be useful because, the same information can be marked in several places in the document which could pose problems

to evaluation methods. In addition, these alternative sets of sentences can be beneficial for machine learning algorithms.

Acknowledgments. The research in this paper is supported by the Arts and Humanities Research Board through the CAST project.

References

1. Marcu, D.: The automatic construction of large-scale corpora for summarization research. In: The 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), Berkeley, CA (1999) 137–144
2. Edmundson, H.P.: New methods in automatic extracting. Journal of the Association for Computing Machinery **16** (1969) 264 – 285
3. Hasler, L., Orăsan, C., Mitkov, R.: Building better corpora for summarisation. In: Proceedings of Corpus Linguistics 2003, Lancaster, UK (2003) 309 – 319
4. Kupiec, J., Pederson, J., Chen, F.: A trainable document summarizer. In: Proceedings of the 18th ACM/SIGIR Annual Conference on Research and Development in Information Retrieval, Seattle (1995) 68 – 73
5. Teufel, S., Moens, M.: Sentence extraction as a classification task. In: Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scallable Text Summarization, Madrid, Spain (1997) 58 – 59
6. Jing, H., McKeown, K.R.: The decomposition of human-written summary sentences. In: Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99), University of Berkeley, CA (1999) 129 – 136
7. Mitchell, T.M.: Machine learning. McGraw-Hill Series in Computer Science. McGraw-Hill (1997)
8. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
9. Byron, D.K., Allen, J.F.: Applying genetic algorithms to pronoun resolution. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence, Orlando, Florida, USA, AAAI Press / The MIT Press (1999) 957
10. Orăsan, C., Evans, R., Mitkov, R.: Enhancing preference-based anaphora resolution with genetic algorithms. In: Proceedings of Natural Language Processing - NLP2000, Springer (2000) 185 – 195
11. Barbu, C.: Genetic algorithms in anaphora resolution. In Antonio Branco, T.M., Mitkov, R., eds.: Proceedings of the 4th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC2002), Lisbon, Portugal (2002) 7 – 12
12. Ke, J., Ogura, M., Wang, W.: Modeling evolution of sound systems with genetic algorithm. Computational Linguistics **29** (2003) 1–18
13. Han, B.: Building a bilingual dictionary with scarce resources: A genetic algorithm approach. In: Proceedings of the Student Research Workshop, the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001), Pittsburgh, USA (2001) 14 – 19
14. Yang, J.J.: Use of Genetic Algorithms for Query Improvement in Information Retrieval Based on a Vector Space Model. PhD thesis, University of Pittsburgh, Pittsburgh, PA (1993)
15. Loose, R.M.: Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: An empirical basis for grammatical rules. Information Processing & Management **32** (1996) 185 – 197

16. Mitchell, M.: An Introduction to Genetic Algorithms. Complex Adaptive Systems. The MIT Press (1996)

Greedy algorithm. Similarity score 0.7064	
S16	Inductive Logic Programming (ILP) is a subfield of Logic Programming and Machine Learning that tries to induce clausal theories from given sets of positive and negative examples.
S24	The two main operations in ILP for modification of a theory are generalization and specialization.
S26	These operations only make sense within a generality order.
S47	Here the concept of a least generalization is important.
S76	In this paper, we give a systematic treatment of the existence and non-existence of least generalizations and greatest specializations, applied to each of these three generality orders.
S88	We survey results obtained by others and also contribute some answers of our own.
S89	For the sake of clarity, we will summarize the results of our survey right at the outset.
S112	Thirdly, we contribute a complete discussion of existence and non-existence of greatest specializations in each of the six ordered languages.
S239	These two different languages and three different quasi-orders give a total of six combinations.
S653	The two main languages are clausal languages and Horn languages.
S654	This gives a total of six different ordered sets .
S655	In this paper, we have given a systematic treatment of the existence or non-existence of least generalizations and greatest specializations in each of these six ordered sets.

Genetic algorithm. Similarity score 0.806	
S28	The three most important generality orders used in ILP are subsumption (also called subsumption), logical implication and implication relative to background knowledge.
S35	Within a generality order, there are two approaches to generalization or specialization.
S86	The combination of three generality orders and two different possible languages of clauses gives a total of six different ordered languages.
S88	We survey results obtained by others and also contribute some answers of our own.
S104	Yet least generalizations relative to function-free background knowledge do not always exist, as we will show in Section 7.
S112	Thirdly, we contribute a complete discussion of existence and non-existence of greatest specializations in each of the six ordered languages.
S113	In particular, we show that any finite set of clauses has a greatest specialization under implication.
S407	It follows from Lemma 1 that G (and hence also G0) cannot contain terms of depth greater than d, nor predicates, functions or constants other than those in S.
S454	If S is a finite set of clauses from C and S contains at least one non-tautologous function-free clause, then there exists a special LGI of S in C.
S653	The two main languages are clausal languages and Horn languages.
S655	In this paper, we have given a systematic treatment of the existence or non-existence of least generalizations and greatest specializations in each of these six ordered sets.
S657	The only remaining open question is the existence or non-existence of a least generalization under implication in C for sets of clauses which all contain function symbols.

Human produced abstract
The main operations in Inductive Logic Programming (ILP) are generalization and specialization, which only make sense in a generality order. In ILP, the three most important generality orders are subsumption, implication and implication relative to background knowledge. The two languages used most often are languages of clauses and languages of only Horn clauses. This gives a total of six different ordered languages. In this paper, we give a systematic treatment of the existence or non-existence of least generalizations and greatest specializations of finite sets of clauses in each of these six ordered sets. We survey results already obtained by others and also contribute some answers of our own. Our main new results are, firstly, the existence of a computable least generalization under implication of every finite set of clauses containing at least one non-tautologous function-free clause (among other, not necessarily function-free clauses). Secondly, we show that such a least generalization need not exist under relative implication, not even if both the set that is to be generalized and the background knowledge are function-free. Thirdly, we give a complete discussion of existence and non-existence of greatest specializations in each of the six ordered languages.

Fig. 3. 2% ideal extracts produced by the greedy algorithm and the genetic algorithm, and the human produced abstract for the text