Evolutionary Algorithms for Definition Extraction

Claudia Borg
Dept. of I.C.S.
University of Malta
claudia.borg@um.edu.mt

Mike Rosner
Dept. of I.C.S.
University of Malta
mike.rosner@um.edu.mt

Gordon Pace
Dept. of Computer Science
University of Malta
gordon.pace@um.edu.mt

Abstract

Books and other text-based learning material contain implicit information which can aid the learner but which usually can only be accessed through a semantic analysis of the text. Definitions of new concepts appearing in the text are one such instance. If extracted and presented to the learner in form of a glossary, they can provide an excellent reference for the study of the main text. One way of extracting definitions is by reading through the text and annotating definitions manually — a tedious and boring job. In this paper, we explore the use of machine learning to extract definitions from non-technical texts, reducing human expert input to a minimum. We report on experiments we have conducted on the use of genetic programming to learn the typical linguistic forms of definitions and a genetic algorithm to learn the relative importance of these forms. Results are very positive, showing the feasibility of exploring further the use of these techniques in definition extraction. The genetic program is able to learn similar rules derived by a human linguistic expert, and the genetic algorithm is able to rank candidate definitions in an order of confidence.

Keywords

Definition Extraction, Genetic Algorithms, Genetic Programming.

1 Introduction

Definitions provide the meaning of terms, giving information which could be useful in several scenarios. In an eLearning context, definitions could be used by a student to assimilate knowledge, and if collected in a glossary, they enable the student to rapidly refer to definitions of keywords and the context in which they can be found. Unfortunately, identifying definitions manually in a large text is a long and tedious job, and should ideally be automated. In texts with strong structuring (stylistic or otherwise), such as technical or medical texts, the automatic identification of definitions is possible through the use of the structure and possibly cue words. For instance, in most mathematical textbooks, definitions are explicitly marked in the text, and usually follow a regular form. In less structured texts, such as programming tutorials, identifying the sentences which are definitions can be much more challenging, since they are typically expressed in a linguistically freer form. In such cases, humans have to comb through the whole text manually to tag definitional sentences.

One way of automating definition extraction is to consult human linguistic experts to identify linguistic forms definitions conform to, usually using either lexical patterns or through specific keywords or cue-phrases contained in the sentence. Once such rules are identified, automatic tools can be applied to find the sentences matching one or more of these forms. This approach has been shown to work with varying results. Technical texts fare better than non-technical ones, where results are usually not of a satisfactory level. Two issues which limit the success of these results are (i) the relative importance of the different linguistic forms is difficult to assess by human experts, and is thus usually ignored; and (ii) coming up with effective linguistic forms which tread the fine line between accepting most of the actual definitions, but not accepting non-definitions, requires time and expertise and can be extremely difficult. Since there typically is a numeric imbalance between definitions and non-definitions in a text, having a slightly over-liberal rule can result in tens or hundreds of wrong positives (non-definitions proposed as definitions), which is clearly undesirable. In the approach we propose, we give a degree of importance (weight) to each linguistic form. Through this technique, one could go further than simple human-engineered linguistic forms — by being able to rank the sentences by how probable the system thinks they are actual definitions. The more a sentence matches against the more important forms, the higher the degree of confidence in its classification as a definition.

In this paper, we explore the use of machine learning techniques, in particular evolutionary algorithms, to enable the learning of sentence classifiers, separating definitions from non-definitions. We have used two separate algorithms for two distinct tasks:

- *Relative importance of linguistic forms:* Given a number of predetermined linguistic forms which definitions may (or usually) conform to, we have used a genetic algorithm to learn their relative importance. Through this technique we enable a more fine-grained filter to select definitions, taking into account multiple rules, but at the same time assigning them different weights before performing the final judgement. We thus benefit from having a ranking mechanism which would indicate a level of confidence in the classification of the definitions. In a semi-automated scenario, it would make the system more usable since a human expert would be presented with the best results first, and results are grouped by 'quality' of the definition.

- *Learning the linguistic forms:* The previous technique assumes that we start off with linguistic forms which are able to match definitions — a task which would typically require human expert input. We incorporated genetic programming techniques to learn such forms automatically by generating different rules in the classification task. Within such a setup it is possible to explore new linguistic structures and test their worthiness automatically against the training data. Rule which are found to be useful in classifying definitions are kept and improved upon to evolve to a better solution.

These two separate techniques are then combined to provide us with a fully automated definition extraction system by first identifying a number of linguistic forms through the use of genetic programming, and then using the genetic algorithm to assign to each rule a degree of importance. The resulting features and their associated weights can then be used by a definition extraction tool which will not only extract candidate definitional sentences, but also rank them according to a level of confidence. The results achieved when combining these two techniques are very promising, and encourage further investigation of these techniques in the task of automatic definition extraction.

In section 2 we give a short overview of definition extraction and the setup of our experiments. In section 3 we describe the results of the genetic algorithm experiment, while in section 4 we present the genetic programming experiments and results achieved. In section 5 we discuss how these two components can be merged into one complete definition extractor, and compare the results to other related work in this area in section 6. We then conclude and discuss future directions in section 7.

2 Definition Extraction

Rule-based approaches to definition extraction tend to use a combination of linguistic information and cue phrases to identify definitions. For instance, in [12, 14] the corpora used are technical texts, where definitions are more likely to be well-structured, and thus easier to identify definitions. Other work attempts definition extraction from eLearning texts [17, 13] and the Internet [6]. Non-technical texts tend to contain definitions which are ambiguous, uncertain or incomplete compared to technical texts.

In our work, we focus on definition extraction from non-technical eLearning English texts in the field of ICT. The corpus consists of a collection of learning objects gathered as part of the LT4eL project [11] which were collected from several tutors in different formats, and standardised in XML format. It is generally recognised that part-of-speech information, which can be extracted automatically from natural language texts is crucial to enable effective discrimination, and the corpus is thus annotated with linguistic information, using the Stanford part-of-speech tagger [15].

The corpus was manually annotated with definitions, to be used as a training set for the definition extraction task. Manually crafted grammars were created in the project to extract definitions, however the results were not satisfactory [1]. From observation it was noted that the structure of definitions does not always follow a regular *genus et differentia* model and different styles of writing and definitions pose a major challenge for the identification of definitions. The solution adopted was to categorise the definitions into different classes, and engineer definition recognisers for each of the classes separately. This reduces the complexity, by attempting to identify a grammar focusing for each type of definition. The types of definitions observed in the LT4eL texts were classified as follows:

1. Is-a: Definitions containing the verb 'to be' as a connector. E.g.: 'A joystick is a small lever used mostly in computer games.'

2. Verb: Definitions containing other verbs as connectors such as 'means', 'is defined' or 'is referred to as'. E.g.: 'the ability to copy any text fragment and to move it as a solid object anywhere within a text, or to another text, usually referred to as cut-and-paste.'

3. Punctuation: Definitions containing punctuation features separating the term being defined and the definition itself. E.g.: 'hardware (the term applied to computers and all the connecting devices like scanners, telephones, and satellites that are tools for information processing and communicating across the globe).'

Three further categories have been identified and used in the LT4eL project, but were not considered for our experiments due to the difficulty of applying machine learning in those instances.

3 Definition Extracting using Genetic Algorithms

Definition extraction is usually based on a set of rules which would have been crafted by a human linguistic expert. The rules would usually contain the discriminating features between definitions and non-definitions, and can be generic (in the form of `Noun Phrase · verb to be · Noun Phrase`) or very specific part-of-speech sequences. Experts usually identify different rules, some of which may be overlapping (that is, a sentence may match more than one rule). Combining such rules can enable more effective definition extraction. At its simplest level, one can adopt the policy which gives preference to sentences which match more of the rules: a sentence matching five rules would be preferred than a sentence matching two rules. However not all rules are equally effective in identifying definitions. Ideally one would want to assign a weight to each rule indicating its relative importance. The setting of these weights can be performed using machine learning techniques.

3.1 Genetic Algorithms

A Genetic Algorithm (GA) [5, 4] is a search technique which emulates natural evolution, attempting to search for an optimal solution to a problem by mimicking natural selection. By simulating a population of individuals (potential solutions) represented as strings, GAs try to evolve better solutions by selecting the best performing individuals (through the use of a *fitness function*), allowing only the best individuals to survive into the next generation through reproduction. This is done using two operations called *crossover* and *mutation*. Crossover takes two individuals (parents), splits them at a random point, and switches them over, thus creating two new individuals (children, offspring). Mutation takes a single individual and modifies it, usually in a random manner. The fitness function measures the performance of each individual[1], which is used by the GA to decide which individuals should be selected for crossover and mutation, and which individuals should be eliminated from the population. This process mimics survival of the fittest, with the better performing individuals being given higher chances of reproduction than poorly performing ones, and thus their winning characteristics are passed on to future generations.

In our work, we have explored the use of a GA to learn the weights to a predetermined set of linguistic rules. These weights will represent the relative importance of the respective rule in its effectiveness at classifying definitions.

3.2 Combining Features

A feature is considered to be a test which, given a sentence s, returns a boolean value stating whether a particular structure, word or linguistic object is present in the sentence — essentially, characteristics that may be present in sentences. These could range from rendering information (bold, italic), to the presence of keywords, or part-of-speech sequences that could identify the linguistic structure of a definition. So, if we take the presence of a bold word to be a feature for definitional sentences, then a sentence containing a bold word is more likely to be a definition than a sentence which does not.

Given a vector of n basic features, $\overline{f} = \langle f_1, \dots f_n \rangle$, and numeric constants, $\overline{\alpha} = \langle \alpha_1, \dots \alpha_n \rangle$, one can define a compound feature combining them in a linear manner:

$$F^{\overline{f}}_{\overline{\alpha}}(s) = \sum_{i=1}^{n} \alpha_i \times f_i(s)$$

Given a sentence, a vector of features and their respective weights, we can thus calculate a numeric value of the sentence by combining the features accordingly. One would also have to identify a threshold value τ such that only sentences scoring higher than this value would be tagged as definitions i.e. s is tagged as a definition if and only if $F^{\overline{f}}_{\overline{\alpha}}(s) \geq \tau$.

[1] The fitness of an individual is the measure of how good this candidate solution is at solving the problem being tackled.

3.3 Learning Weights

We have used a GA to identify a good set of weights and the threshold value for a given set of features. Each individual in the population of the genetic algorithm is represented as the vector of numeric weights. Crossover between individuals simply consists of splitting the vector of the two parents at a random position, and joining the parts, thereby creating two new individuals for the next generation.

What the GA learns is determined by the fitness function, which, given an individual, returns a score of how 'good' the individual is. We have used a corpus of definitions and non-definitions to evaluate the performance of each individual. The fitness function takes an individual (vector of weights) and runs through the whole corpus using the combined feature function and calculates how many definitions are correctly classified, and how many are incorrectly tagged as non-definitions. Similarly, we compute the values for the non-definitional sentences. Through these figures we are then able to extract precision, recall and f-measure. We have run the GA using these different measures as the fitness function.

The choice of threshold is obviously crucial to the value returned by the fitness function. One option was to set the threshold to a fixed value for the whole population (say, at zero). However, it was noted that given an individual one can actually compute an optimal value for the threshold with respect to the corpus using an efficient (linear) algorithm. Another option was to include the threshold as part of the individual's chromosome. However, this would have serious implications on the effectiveness of the learning unless the crossover function is defined in a more careful manner, since during crossover one would mix-and-match weights of individuals with different thresholds. Combining two good individuals would typically result in a non-effective one in this manner. We opted not to explore this option.

Two experiments were run, one with a fixed threshold value of zero, and another using optimal (individual specific) thresholds, with the latter achieving far better results.

3.4 Experimental Results

The GA experiments focused on the 'is-a' category, where we had 111 definitions and 21,122 non-definitional sentences. Several experiments were carried out, using different techniques within the algorithm mechanics. The best selection algorithm was SUS with sigma scaling [10]. Here we present a summary of the best and most interesting results of this work.

During the set up of the GA, we used a simple set of ten features which were hand-coded and inputted into the GA for it to learn their relative importance. Following is the set of features used:

1. contains the verb "to be"

2. has sequence "IS A" ("to be" followed by a determiner)

3. has sequence “FW IS” (FW is a tag indicating a foreign word - in the example “The process of bringing up the operating system is called booting”, booting is tagged as an FW.)
4. has possessive pronoun (I, we, you, they, my, your, it)
5. has punctuation mark in the middle of the sentence (such as a hyphen or colon)
6. has a marked term (keyword)
7. has rendering (italic, bold)
8. has a chunk marked as an organisation
9. has a chunk marked as a person
10. has a chunk marked as a location

These features were purposely simplistic when compared to the manually crafted rules in the LT4eL project for definition extraction. This enabled us to analyse the relative weights assigned and to be able to allow more focus on the algorithmic aspects of the GA. These features were used throughout all the experiments discussed in this section.

Table 1: *Results for best experiments*

Method	**F-measure**	**Precision**	**Recall**
Experiment 1	**0.57**	0.62	0.52
Experiment 1a	0.62	**0.70**	0.42
Experiment 1b	0.54	0.46	**0.56**
Experiment 2	**0.57**	0.64	0.50
Experiment 3	**0.54**	0.59	0.50

Table 1 presents the results achieved by the best performing runs, indicating the f-measure, precision and recall achieved by assigning the weights learnt to the set of features. The best runs achieved an f-measure of 57%, with the runner-up achieving 54%. Since we used f-measure as the basis of measuring the weights' effectiveness in classifying definitions, we were also able to influence f-measure to favour precision or recall according to the setting of the alpha value. Experiments 1a and 1b show the results for favouring precision and recall respectively.

Using a small set of simple features, the GA has managed to obtain positive results, especially when comparing to the manually crafted grammars in LT4eL. We have increased precision from 17% to 62%, whilst maintain recall over 50%. Further improvement would probably be achieved had we to include more rules from the manually crafted grammar as part of our set of features.

The possibility of influencing the learning of weights to favour precision or recall is considered a positive facility in this experiment, since the end use of the definition extraction tool could require different settings. In a fully automatic system, precision might be given more importance, whilst in a semi-automatic system, recall is more important since a human expert will verify the correctness of the candidate sentences.

```
feature         ::=  simplefeature
                 |   simplefeature & feature

simplefeature   ::=  lobj
                 |   emptystring
                 |   any
                 |   simplefeature ?
                 |   simplefeature *
                 |   simplefeature . simplefeature
                 |   simplefeature + simplefeature
```

Fig. 1: *Specification of the representation of individuals*

4 Feature Extraction using Genetic Programming

The main bottleneck of using the GA as discussed in the previous section is that the linguistic rules have to be identified by a human expert. From the LT4eL experience it was clear that linguistic experts were needed to identify complex rules which non-experts would not have identified. The rules identified by experts are typically expressed as complex grammars or regular expressions ranging over parts-of-speech. In this section we present another experimental setup we have used to explore the use of machine learning techniques for the automated identification of linguistic rules.

4.1 Linguistic Rules

Recall that linguistic rules are objects which given a sentence, return a boolean value, depending on whether or not the sentence matched the rule. One way of expressing such rules is through the use of regular expressions, e.g. `noun·is·a·noun`. These regular expressions would range over the grammar shown in figure 4.1.

Note that the basic elements of the regular expression are simple linguistic objects (with no structure). Note also that to enable more complex rules, we allow not only the usual regular expression operators (optional inclusion, repetition, catenation and choice), but also allow the conjunction of regular expressions at the top most level (thus controlling the computational complexity of matching the regular expression).

The framing of basic features as instances of this language of regular expressions, enables us to formulate the task of the learning algorithm as that of learning an instance of this language (of regular expressions) which is effective when used for definition extraction.

For the choice of linguistic objects, we chose to either use specific part-of-speech tags such as NN (noun, common, singular or mass) or to generalise these tags into one class and refer to them as nouns.

4.2 Genetic Programming

Genetic programs (GP) are another form of evolutionary algorithms introduced by [8] whose aim is that of automatically learning instances of a language, typically computer programs, automatically. The algorithm is very similar to GAs in structure — it is a search optimisation technique, exploring different possible solutions to a problem. Similarly to a GA, this technique uses crossover and mutation to evolve new individuals, and a fitness function to test the strength of the individual. One of the main differences between the two techniques is that unlike GAs, a GP uses tree representation to represent the individual.

Several of the definition extraction tasks tend to use rules made of part-of-speech information, which is generally arrived to through linguistic expertise or through observation of definitional sentences and their linguistic structure. In the process of creating such rules it is usually not very clear as how to best tweak a rule for better performance. Thus, an experimental setup which would create rules automatically and test them upon an annotated corpus is desirable. When a rule created is able to match correctly a sentence, it is kept as a potentially good rule to use in a definition extraction tool. A GP is an ideal experiment for this task as it facilitates the process of rule discovery and tests their effectiveness through the evolutionary process.

Since the evolutionary process is based on matching sentences against the rules created, we have also used f-measure as the fitness metric to determine whether a rule (an individual) is a good possibility or not. Those rules which have a higher f-measure will be kept by the GP so as to explore similar possibilities.

4.3 Experimental Results

Experiments using the GP delved into the three categories identified in section 2, that is the is-a, verb and punctuation categories. For each category, several experiments were run, each of which resulting in different rules (albeit at times quite similar). Experiments also tested the inclusion of different linguistic objects by either focusing on specific POS tags, or by generalising the particular category, say to include all nouns. In the case of the verb category, some of the experiments focused on the POS tags, while others included certain words such as ‘known’, ‘define’ and similar words typically found in definitions in this category.

Table 2 shows a summary of the best results achieved in the different categories where the GP was applied. The experiments were run with different population size ranging from 200 to 1,000 individuals. Most of the experiments converged within 100 generations, and at times as early as 30 generations. The selection of the individuals to survive to the next generation used elitism (which copies the best individuals of the population into the next generation as is), and selecting the remaining individuals for crossover using the stochastic universal sampling algorithm [10]. Further details about the experimental setup can be found in [2].

The GP was able to learn at times rather simple rules such as `noun·is·a·noun`. The rules learnt for each category by the different experiments were usually similar in structure and content. However in certain runs the rules represented by the individuals gave better results. In the is-a category, the average f-measure obtained was around 25%, with one run managing to produce a slightly different rule achieving 28% f-measure. In the verb category it was noticed using part-of-speech categories was not sufficient, and that the use of keywords, such as ‘know’, ‘define’, and ‘call’, was necessary to achieve good results. In the punctuation category we observed that results were achieved easily primarily due to a smaller search space when compared to the other categories.

Table 2: *Summary of results*

Category	**F-measure**	**Precision**	**Recall**
Is-a	0.28	0.22	0.39
Verb	0.20	0.14	0.33
Punctuation	0.30	0.25	0.36

5 Combining the Experiments

The two experiments described above were so far isolated, each one with a particular purpose. The challenge towards which we worked is to have a fully automated definition extraction tool which is easily adaptable to different domains and which ranks candidate definitions according to some level of confidence. In this section we describe how these two separate experiments were combined together towards a fully automated definition extraction tool. In figure 3 we see the different phases of the definition extraction process. Phase one is the creation of an annotated training set and is not dealt with in this work. Given an annotated corpus with definitions, one can then move onto phase two where the GP is applied to learn useful simple features which can be used to distinguish definitions from non-definitions. In phase three the GA is then used to learn weights for the rules learnt by the GP. Using the rules and weights, one can incorporate all this in a definition classification tool in phase four. In this section we present the results achieved from combining phase two and three together.

For the purpose of combining the two phases, we used the best rules learnt by ten different GP experiments in the is-a category. These individuals were used by the GA to learn their respective weights. The set up is shown in figure 2 where the final result is a set of rules in the is-a category together with their allocated weights indicating the level of effectiveness each weight has. As shown in the previous section, the rules the GP learnt without the application of the weights resulted at best in f-measure being 28%. Once weights were learnt and applied to the definition extraction tool, this increased to 68% f-measure. This improvement shows that learning weights is useful to the classification task since it does matter which rule is actually carrying out the classification of sentences.

Further analysis show that the f-measure is resulting from a 100% precision and a 51% recall. This means that by combining the rules learnt and their associated weights, we succeeded in classifying just over half of the annotated definitions, without classifying any incorrect definitions. There are several factors behind these results:

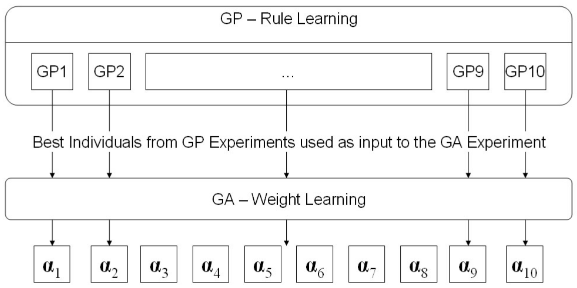
GP – Rule Learning
GP1
GP2
...
GP9
GP10
Best Individuals from GP Experiments used as input to the GA Experiment
GA – Weight Learning
α1 α2 α3 α4 α5 α6 α7 α8 α9 α10

Fig. 2: *Combining the two experiments*

1. This experiment was carried out on only one corpus, so the rules learnt together with their respective weights, were specific to the corpus used. Achieving such a good result is only indicative that as in any machine learning process, the two algorithms were able to learn rules and weights specific to our corpus.

2. The recall of 51% represents definitions for which the genetic program did not learn rules for. Since these algorithms are searching for solutions in an automatic manner without expert feedback, it is the case that not all possible rules are explored. This can be tackled by including rules from more experiments or by having direct feedback from a linguistic expert (say, injection of good humanly crafted rules into the population).

Notwithstanding the conditions under which they were achieved, the results are very promising.

6 Discussion and Related Work

Although the results achieved so far are promising and encourage further investigation of these techniques, it is difficult to provide a fair and just comparison to other techniques. One of the main reasons is that an evaluation using an unseen corpus is required to have a more realistic view of the results achieved using these techniques. To our knowledge there is no other work in definition extraction using evolutionary algorithms to which our results can be directly compared to.

However, there are various attempts at definition extraction using different techniques. DEFINDER [12] is a rule-based system which extracts definitions from technical medical texts so that these can later be used in a dictionary. The rules are primarily based on cue-phrases such as “is called a”, with the initial set of candidate sentences being filtered out through the use of POS rules and noun phrase chunking. They manage to obtain a precision of 87% and a recall of 74%. Definition extraction is also considered to extract the semantic relations present in definitions. In [9], they apply lexico-syntactic patterns in addition to cue phrases, focusing on hypernym and synonym relations in sentences. They obtain 66% precision and 36% recall.

Work carried out in [14], applies valency frames to capture definitional sentences achieving an average of 34% precision and 70% recall across the rules created. A German corpus consisting of legal decisions is used in [16] to extract definitions. They analyse the structure of definitions in this domain, and observe that the German word `dann` can be used as a signal word indicating that a sentence is a definition. There is no equivalent term in English. The rules are crafted manually through observation, and achieve an average of 46% precision. When only the most effective rules are used, precision increases to over 70%, however recall is not discussed since the corpus is not annotated with definitions. Extraction of definitions from eLearning texts is attempted for the Slavic group of languages in [13], using noun phrase chunking and phrase structure as the potential identifying features in definitions. The best results are achieved for the Czech language with precision at 22% and recall at 46%.

Research in general seems to point out to the need of going beyond rule-based techniques, and trying out machine learning to improve definition extraction. Definitions extracted from the Dutch Wikipedia from

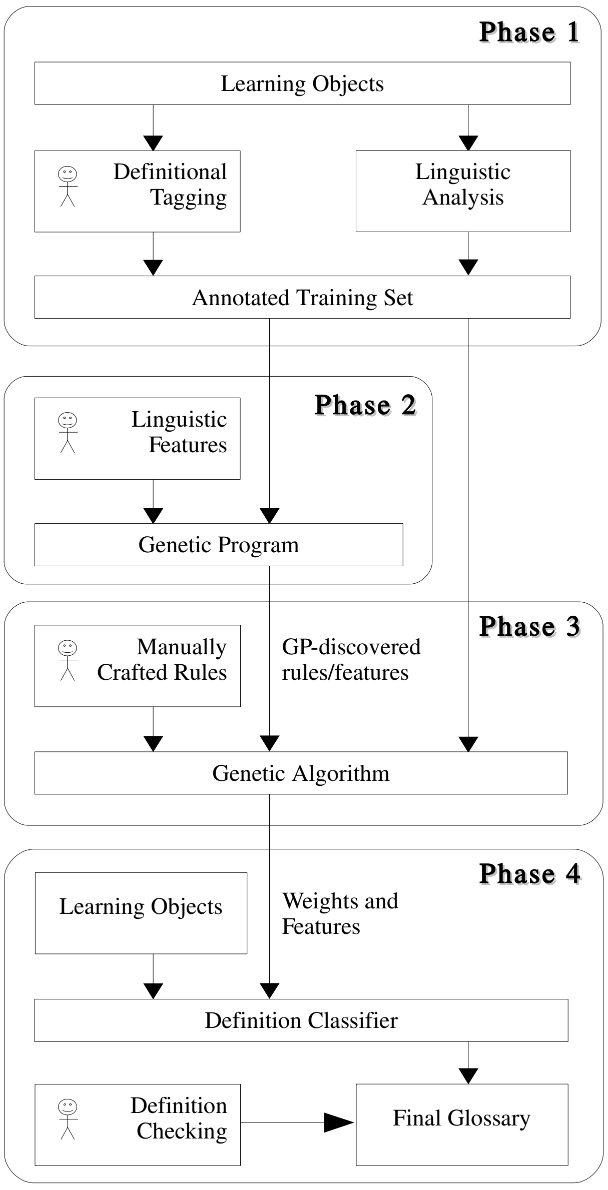

Fig. 3: *Phases of definition extraction*

medical articles in [3] first use a rule-based approach using cue-phrases, but further improve their extraction process by using Naive Bayes, maximum entropy and SVNs. As part of their feature set they include sentence positioning, a feature which cannot be applied to other types of corpora. The best result is from applying maximum entropy, achieving 92% accuracy. Similar experiments by [17] on an eLearning corpus obtain 88% accuracy, with the difference in result being due to the type and structure of the corpus used. Similarly [7] obtain an accuracy of 85% using a Balanced Random Forest on an eLearning corpus. These techniques all share the similarity in having improved considerably the results of manually crafted grammars when applying machine learning techniques.

7 Future Directions

In this paper, we have presented a methodology for the use of evolutionary algorithms to create sentence discriminators for definition extraction. We have shown how GPs can be used to learn effective linguistic rules, which can then be combined together using weights learnt through the use of a GA. The overall system can, with very little human input, automatically identify definitions in non-technical texts in a very effective manner. Using our approach we have managed to learn rules similar to the manually crafted ones by the human expert in the LT4eL project, and further associate them with weights to identify the definitions in non-technical texts — all performed in an automated fashion. One of the major strong points of the approach is that the (expensive) learning phases is performed once, and the resulting definition discriminator is very efficient, making it viable to be included in other applications.

The final experiment of using both techniques for definition extraction gave surprising results, managing to identify only definitions, achieving a 100% precision, albeit having identified rules to capture only half of the definitional set of sentences. This result is certainly encouraging when considering that the process is fully automated.

There are various directions we plan to explore in the future. Our experiments would need to be evaluated further, experimenting with other corpora in different domains. For instance, medical texts contain several terms which a part-of-speech tagger might not recognise and would tag as 'foreign word'. Thus the rules learnt for our eLearning corpus might not necessarily apply for a medical corpus.

We also intend to evaluate the definition extraction tool over an unseen corpus. Such an evaluation might show that the rules learnt by the GP are not generic enough to cover unseen definitions, a result which is common in such machine learning techniques. It would be ideal to have some form of feedback loop from an expert to the learning algorithm to integrate new knowledge gained over unseen corpora.

We plan to explore and assess the use of weights to go beyond a crisp discriminator, and interpret the results as a fuzzy discriminator, associating a degree of confidence with each sentence, thus enabling us to rank definitions according to how sure the system is that it is a definition. This is crucial if the definitions discovered are to be vetted by a human operator.

Finally, we plan to extend the use of GP to learn rules in an iterative manner. After each iteration of the experiment, the sentences for which it learnt rules are removed from the training corpus, and the experiment repeated. In this way we would be reducing the search space, and forcing the GP to learn new rules. It might be the case that the GP does not learn certain rules as they would classify to many non-definitions to simply capture few definitions. However, by carrying out such an experiment we might be able to learn rules which cover the search space better, and at the same time identify those definitions for which it is difficult to define rules which provide acceptable results.

References

[1] C. Borg. Discovering grammar rules for Automatic Extraction of Definitions. In *Doctoral Consortium at the Eurolan Summer School 2007, Iasi, Romania.*, pages 61–68, 2007.

[2] C. Borg. Automatic Definition Extraction Using Evolutionary Algorithms. Master's thesis, University of Malta, 2009.

[3] I. Fahmi and G. Bouma. Learning to Identify Definitions using Syntactic Features. In *Workshop of Learning Structured Information in Natural Language Applications, EACL, Italy*, 2006.

[4] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[5] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[6] J. L. Klavans, S. Popper, and R. Passonneau. Tackling the internet glossary glut: Automatic extraction and evaluation of genus phrases. In *SIGIR'03 Workshop on Semantic Web*, 2003.

[7] Ł. Kobyliński and A. Przepiórkowski. Definition Extraction with Balanced Random Forests. In *proceedings of GoTAL*, 2008.

[8] J. R. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Selection.* MIT Press, Cambridge, MA, 1992.

[9] V. Malaisé, P. Zweigenbaum, and B. Bachimont. Detecting semantic relations between terms in definitions. In *COLING CompuTerm 2004: 3rd International Workshop on Computational Terminology*, pages 55–62, 2004.

[10] M. Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, 1998.

[11] P. Monachesi, L. Lemnitzer, and K. Simov. Language Technology for eLearning. In *First European Conference on Technology Enhanced Learning*, 2007.

[12] S. Muresan and J. L. Klavans. A method for automatically building and evaluating dictionary resources. In *Proceedings of the Language Resources and Evaluation Conference*, 2002.

[13] A. Przepiórkowski, Ł. Degórski, M. Spousta, K. Simov, P. Osenova, L. Lemnitzer, V. Kubon, and B. Wójtowicz. Towards the automatic extraction of definitions in Slavic. In *Proceedings of the BSNLP workshop at ACL*, 2007.

[14] A. Storrer and S. Wellinghoff. Automated detection and annotation of term definitions in german text corpora. In *Language Resources and Evaluation Conference*, 2006.

[15] K. Toutanova and C. D. Manning. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), Hong Kong*, 2000.

[16] S. Walter and M. Pinkal. Automatic Extraction of Definitions from German Court Decisions. In *Workshop on Information Extraction Beyond The Document*, pages 20–28, 2006.

[17] E. Westerhout and P. Monachesi. Extracting of Dutch Definitory Contexts for elearning purposes. In *CLIN*, 2007.