Towards Automatic Extraction of Definitions

Claudia Borg, Mike Rosner, and Gordon Pace

University of Malta

Abstract. Definition Extraction can be useful for the creation of glossaries and in Question Answering Systems. It is a tedious task to extract such sentences manually, and thus an automatic system is desirable. In this work we will review some attempts at rule-based approaches and discuss their results. We will then propose a novel experiment involving Genetic Programming and Genetic Algorithms, aimed at assisting the discovery of grammar rules which can be used for the task of Definition Extraction.

1 Introduction

A definition is a description of a term which states its meaning, or describes its concept. Definitions are helpful because they facilitate the understanding of new terms. The extraction of definitions from text can be useful in various scenarios, including the automatic creation of glossaries for building of dictionaries and in question answering systems. In this work, we will focus on eLearning, where definitions can help learners conceptualise new terms and help towards the understanding of new concepts encountered in learning material.

eLearning is the process of acquiring knowledge through electronic aids, by providing access to materials that will enable them to learn a particular task. Learning material, also referred to as learning objects (LOs) normally contain various definitions which describe the concepts required by students as part of the learning process. In eLearning, the tutor can direct the learning process through a Learning Management System (LMS), where LOs can be presented to the student according to the tutor's direction. The tutor can also use LMS to add new content, create courses, structure layout and presentation of courses and monitor student performance.

An LMS can be enhanced by introducing definition extraction to its functionality. The LMS could suggest definitions contained in the content to the tutor, and the tutor can use such definitions to provide more information to the learner. The tutor in such a situation has the opportunity to act as an expert, and approve or discard definitions proposed by the system.

The task of definition extraction is a very challenging one. We are trying to identify sentences that contain knowledge about a specific term, which could then be used in applications mentioned above. What's more, we are attempting to identify sentences which *define* a term, rather than simply describe it vaguely or compare it to other terms. As in many NLP tasks, the problem could be alleviated by including more information about the text, such as part-of-speech tagging and morphological analysis. Rather than trying to identify arbitrary definitions, one can start by classifying them into different categories, and definition extraction can then be attempted on each category separately to achieve a divide-and-conquer approach to the task. Once all definitions contain the linguistic features and definition categories, definition extraction is more attainable.

In section 2 we will review the work that has been carried out in attempting definition extraction. We will then propose an experiment in section 3 which combines genetic algorithms (GA) and genetic programming (GP) to try and discover grammars that could identify definitions present in learning objects. The outcome of this work is to evaluate the use of such machine learning techniques and their results in learning restricted grammars. The grammars developed through these experiments can then be applied to rule-based techniques to extract definitions. The results of the GP and the GA will be used to discover features which identify certain definitions with a high rate of accuracy, but also other features to classify the less clearcut definitions using the features in a combined manner. In section 4 we will describe work using GAs and GPs in similar areas, and discuss which features of such work can be applicable to our proposed techniques.

This work is done in collaboration with an EU-funded FP6 project LT4eL[1] (Language Technologies for Learning). The project is described in more detail in [BR07] and [MLS07], and aims at enhancing LMS by using Language Technologies and Semantic Knowledge.

2 Background

2.1 Definition Extraction

Definition extraction is an important task in NLP, and is usually considered as a subtask of information extraction, automatic creation of glossaries, question answering systems.

Work carried out on automatic creation of glossaries usually tends to be rule-based, taking into consideration mainly part-of-speech as the main linguistic feature. Park et *al.* [PBB02] propose a system whereby glossary candidates are presented to an expert in the relevant domain to be approved and then made available through an API to an application system. In their work, they concentrate on detecting the terms and their glosses rather than full definitions. Glosses present a summary of the meaning, usually giving the full form of an abbreviation or a variant of the term. They also deal only with technical texts, where glosses are normally well structured.

Their pipeline architecture uses several tools, including POS tagging and morphological analysis, with each tool providing additional annotations. The glossary extraction algorithm first looks at the structure of glossary items. They identify POS structures for noun phrases or verbs, which is used to identify possible glosses. These are described as a cascade of finite-state transducers,

[1] www.lt4el.eu

which are easy to extend and re-use even across different languages. They also state that it is difficult to simply look at POS sequences to identify glosses, as many other non-gloss items would have the same sequence. To overcome this problem, rules are applied to discard certain forms from the candidate set. These include person and place names, special tokens such as URLs, words containing symbols (except for hyphens and dashes) and candidates having more than 6 words.

Variants are identified, grouped and one is set at the canonical form, others listed as variants (including misspelt items and abbreviations). Finally all glossary candidates are ranked and presented to the expert.In the evaluation of their work, three human experts accepted 228 (76%), 217 (72%), and 203 (68%) out of the top 300 as valid glossary items. Inter-annotater agreement is not discussed in this evaluation.

Klavans and Muresan [KM00] propose a system, Definder, to extract definitions from technical, medical texts. Corpora used are consumer-oriented text, were a medical term is explained in general language words in order to paraphrase the equivalent specialised terms. The aim for their system is to be able to extract definitions that can then be fed into a dictionary. Their approach takes NLP techniques into consideration in order to be able to identify definitions. Synonyms are also considered as definitions. They point out that the structure of definitions might not follow the *genus et differentia* model and that the different styles of writing can be a challenge for the automatic identification of definitions.

Definder first identifies candidate phrases by looking for particular cue-phrases such as “is called a”, “is the term for”, “is defined as”, or a set of punctuation marks which are deemed important for this task (namely -, (,), :). A finite state grammar is then applied to extract the definitions. The system uses part-of-speech and noun phrase chunking to help with the identification process. In order to improve results, the Definder uses statistical information from a grammar analysis module. They claim that this will take into account the styles for writing of definitions (apposition, relative clauses, anaphora).

In this work we see that the automatic identification of definitions is mainly based on the primary identification of certain phrases, and then further filtered through certain rules that reinforce a sentence being a definition (such as its POS structure). Although there is no attempt at automatically trying to identify the possible features that could result in capturing definitions, the use of statistic methods on parsing of grammars can be conceived as such a possibility.

In further work Klavans et *al.* [KPP03] look at the Internet as a corpus, focusing mainly on large government websites, trying to identify definitions by genus to extract conceptual relations for ontology building. In this task, several problems are identified, both in format and in content. Definitions can be ambiguous, uncertain or incomplete. They are also being derived from heterogeneous document sources. Another problem encountered is that the Internet is a dynamic corpus, and websites could change their information over time. An interesting discussion is presented in how to evaluate a definition extractor, proposing a Gold Standard for such a type of evaluation, based on qualitative evaluation apart from the standard quantitative metrics.

Liu et *al.* [LCN03] are interested in definition extraction of concepts for learning purposes. Their strategy to assist learning is to present learners with definitions of concepts, and the sub-topics or salient concepts of the original topic. Their system queries search engines with a concept, and the top 100 ranked results are retrieved. In order to discover salient and sub-topics, they look at layout information presented in the html tags for features such as headings, bold and italic. A rule-based approach is then applied to filter out items which are generally also highlighted in webpages (such as company names, URLs, lengthy descriptions). Further filtering is applied through stopword removal, taking frequency into consideration and ranking the proposed salient or sub-topics.

Definition extraction is then attempted for the concepts and sub-concepts identified in the first phase of their work. The identification is carried out through rule-based patterns, (e.g. {concept} {refers to|satisfies}. . .). Webpages containing definitions are attached to the concept, and presented to the user in ranked order. The ranking is based on how many concepts/definitions are present in a webpage (the more being available, the higher the ranking as it is considered more informative).

Liu et *al.* also propose a way of dealing with ambiguity, where the term being learned is too generic and can appear in different context (e.g. classification - library classification, product classification, data mining classification). Again, to differ between different context, the term is allocated a parent topic, and webpages are searched for the occurrence of the concept present in certain html layout structures. Then a hierarchical structure of topics is built according to the findings. Mutual reinforcement is also used to provide further evidence of the hierarchy built, by further searching for sub-topics under a particular concept. This generally would result in finding information about other salient topics under that same concept, which thus continues to re-ensure that the sub-topics are related and belong to the parent concept.

In this work we note that the definition extraction phase is preceded by a useful phase of concept identification. Thus the task of definition extraction itself is simplified, as the search made is for a particular definition, and not all definitions contained in a given text. This might result in definitions of other concepts being lost in already parsed documents. The work also does not make any use of linguistic information, since at their level of processing simple pattern matching on particular keywords is sufficient. The web is a very large corpus and thus can provide many documents/definitions. However, this can also be a disadvantage effecting the quality of definitions found. This is partially surpassed by providing several resulting definitions, and documents containing more definitions (and thus might be more authoritative) are presented first. However, quality of definitions is important in any learning phase, and can determine the learner's understanding of a concept.

Storrer and Wellinghoff [SW06] report work on definition classification based on 19 primary verbs, specified in valency frames. These frames indicate what arguments a verb takes, including object, subject, position, prepositions etc. This frame can be used to match the structure of a sentence containing one of the specified 19 verbs. Thus definitions are extracted by using the valency frames specified for the defining verbs. The approach presented in the paper is a rule-based expert driven, with all information being provided by human experts (valency frame, definition categorisation). This is possible because they are looking at technical texts, where definitions are well-structured, frequently matching more crisp rules.

2.2 Statistical Learning Methods

Fahmi and Bouma [Fah06] tackle the problem of definition extraction using an incremental approach, starting with individual words, then adding syntactic features etc. They look at the potential definition sentences that fall into our first category (containing the verb to be) from a Dutch corpus of medical articles extracted from Wikipedia. These sentences are manually annotated as definitions, non-definitions and undecided, and this corpus of sentences is used as their training and evaluation data for the experiments carried out. They identify several attributes that could be of importance to the experiments, namely text properties, sentence position, syntactic properties and named entity classes. Learning-based methods are then used to identify which of these features, or combination of, would provide the best results. These feature combinations can also be considered for the experiments described above.

3 Proposed Solution

3.1 Experiment One: Genetic Algorithm

The availability of manually annotated definitions places us in a favourable situation to attempt learning how to recognize definitions. Given that a corpus contains both a set of definitions and a (usually larger) set of non-definitions, an attempt to learn the importance of features present in definitions is possible. A feature can be seen as a description of characteristics that can help us identify a definition.

A genetic algorithm [Hol75,Gol89] is an ideal technique that can be used to learn the importance of the features that can recognise definitions. This can be done by assigning weights to each feature and allowing the algorithm to adjust the weights according to the performance. It also makes it ideal to run the GA on the separate categories of definitions identified (as described in [BR07]), so that the results can be directed to one given situation at a time.

A feature can be described as a function which given a sentence will return a score. As an example, one feature could describe a part-of-speech sequence that might capture a definition (e.g. DT→NN→VBZ→DT→NN→IN→NNS). It would also be possible to produce more generic features such as “Contains the verb to be”. The score output from the function will indicate how the sentence rates when the function is applied. The score can be either simply 1 or

0 to indicate a match or non-match, or some numeric value to indicate a level of matching. The latter situation might produce better results by giving more flexibility to the scoring function. For instance, if a sentence would not match the above feature by one pos tag, then the score would be higher than that of a sentence which does not match a single pos tag. If we have n features, we would thus have n functions f_1 to f_n, each of which takes a sentence and returns a numeric score.

It is important to note that the role of the Genetic Algorithm is not to learn new features but rather to learn the effectiveness of features, as classifiers of definitions.

Before starting the experiment, a predefined set of features will be adopted from the current set of definitions. This collection of features will remain static throughout the experiment. Furthermore, for each such feature we will also have access to a function which will return a score for a given sentence when applied to that feature.

A gene will be a list of length equal to the number of predefined features of numbers. Thus, the ith gene ($1 \leq i \leq \mathit{populationsize}$) will have the following structure:

$$g_i = \langle \alpha_{i,1}, \alpha_{i,2} \ldots \alpha_{i,n} \rangle$$

Note that n corresponds to the number of predefined features. The interpretation of the gene is a list of weights which will be applied to the output of the predefined features. Thus, $\alpha_{i,1}$ is the weight that the gene g_i would assign to the first feature. Such a gene will therefore return a score to a given sentence s as follows:

$$\mathit{score}_i(s) = \sum_{j=1}^{n} f_j(s) \times \alpha_{i,j}$$

The initial population will consist of genes that contain random weights assigned to each feature. The interpretation of the gene is a function that when applied to a sentence gives the summation of the feature-function scores multiplied by their weights.

The fitness function will take an individual and produce a score according to its performance. The score will be calculated by applying the gene to both positive and negative examples and will be judged according to how the gene is able to separate the two sets of data from each other.

Crossover and mutation will be carried out in the ‘traditional’ way of Genetic Algorithms. Crossover will take two individuals, bisect them at a random position and create two new children by interchanging the parts of the parents. Mutation will take a random position in the gene and change its value. If the children perform better than the parents, they will substitute them.

Once the population converges, an expected outcome of this experiment is the interpretation of the best gene. The weights produced would give the clearest separating threshold between definitions and non-definitions. It will also allow us to identify which of the features in the predefined feature set are most important, due to a larger weight.

3.2 Experiment two: Genetic Programming

Genetic Programming [Koz92] is a technique that uses Genetic Algorithms at its underlying basis, in that it is an evolutionary algorithm. However, Genetic Programming is an optimization problem that evolves simple programs. The main difference between the two techniques is the representation of the population and how the operations of crossover and mutation are carried out. The members of the population are parse trees, usually of computer programs, which are evaluated by the fitness function through their execution. Crossover and mutation are carried out on subtrees, ensuring that the resulting tree would still be of the correct structure.

Whereas the scope of the previous experiment was to learn which are the best performing features from a set of predefined ones for the task of definition extraction, this experiment aims at identifying new features. The initial population will consist of features which were given higher weights by the GA. These features will be translated into parse trees and the score of the fitness function will indicate how well the individual can discriminate between definitions and non-definitions.

The choice of what type of structure we are trying to learn is a determining factor to the complexity of the search space. In our application, two possible options could be regular languages (in the form of regular expressions) or context-free languages (in the form of context-free grammars), the latter having a large search space than the former. Through observations and current work with lxtransduce, regular expressions (extended with a few constructs) would be sufficient to produce expressions that would correctly identify definitions in most cases.

Some of the features used in the first experiment can be used to inject an initial population into the Genetic Program. The selection can be made based on the weights learned by the GA and translating those features into extended regular expressions. The extensions that are being considered are conjunction (possibly only at the top level), negation and operators such as *contains sub-expression*. Note that some of these can already be expressed as regular expressions, however, introducing them as a new single operator helps the genetic program learn them faster.

The population will evolve with the support a fitness function in order to select those individuals for mating. The fitness function can apply the extended regular expressions on the given training set and then use measurements such as precision and recall over the captured definitions. Such measurements can indicate the performance of the individuals and will allow us to fine-tune the GP according to the focus of the experiment (where one could emphasis on a high percentage for one measurement at a time, or take an average for both). This flexibility will also allow us to have different results in the various runs of the experiment, where, for instance, in one we could try to learn over-approximations whereas in an other we can learn an under-approximation.

Crossover will take two trees and create two new children by exchanging nodes with similar structure. If an offspring is able to parse correctly one definition, it survives into the next generation, otherwise it is discarded. Parents would normally also be retained in the population, since we would not want to lose the good individuals (it is not obvious that their offspring would have the same capability of identifying definitions).

Mutation would take an individual and select at random one node. If that node is a leaf, it is randomly replaced by a terminal symbol. If it is an internal node, it is randomly replaced by one of the allowed operators. Once again, the new tree is allowed to survive to the next generation only if it is able to capture at least one definition.

Once the Genetic Program converges, we expect to have new expressions that would capture some aspects of a definition. The application of this program will allow us to extend our current set of grammar rules by deriving new rules from the above operations. Although we do not expect the genetic program to learn exact rules, it will help towards the discovery of new rules which might have been overlooked, and thus helping towards a more complete grammar for definition extraction.

The GP will also allow the flexibility of running this experiment separately for each of the categories of the definitions as identified in section 3.2. This means that the new features being learned will be restricted to one category at a time.

3.3 Combining the two experiments

The role of this work is to develop techniques to extract definitions. The two experiments are independent of each other. The GA takes a set of features and assigns a weight to each feature, whereas the GP learns new features through the evolution of the population of extended regular expressions. We can combine the two experiments by migrating the new features learned by the GP to augment the feature set which is used in the GA.

In the final definition extractor one can start by checking whether a given sentence can be confidently classified as a definition or not by using the features one may learn by running the GP trying to find features which are strict over- or under-approximations. One would then run the weighted sum and threshold as learned by the GA based on the features we manually identified and others that the GP may have learned. Clearly the training of the GA would have to be done on a subset of the training set, removing the confidently classified non/definitions. We believe that this approach will improve the quality of the definition identifier.

4 Literature Review

Identifying grammars for the task of definition extraction can be related to the area of Grammatical Inference. In this field, research attempts to use machine learning techniques to learn grammars and language from data. A variety of applications can be identified including syntactic pattern recognition, computational biology, natural language acquisition, data mining and knowledge discovery. Genetic algorithms (GA), and less frequently, genetic programming (GP) are two machine learning techniques which have been applied to grammatical inference. This section will review literature in this area and identify methods or ideas which could be applied to our problem.

Genetic Algorithms

Lankhorst [Lan94] describes a genetic algorithm used to infer context-free grammars from legal and illegal examples of a language. A discussion is presented on why GAs are expected to work in the application of Grammatical Inference. The schema theorem states that schemas occurring in the higher scoring individuals will tend to occur more frequently in following generations. This feature, also referred to as the *building blocks hypothesis*, is the motivation for applying GAs to grammatical inference. New, possibly better performing, grammar rules can be discovered by combining parts of different grammar rules.

The paper goes on to discuss the representation of the individuals in a population, between a binary and a high-level representation. It is argued that a bit string can represent many more schemata than higher level representations, yielding to a better coverage of the search space. A bit representation is chosen, with the lower order bits encoding grammar rules on the right hand side of the rule, whereas higher order bits are encoding the left-hand side symbol.

Selection is based on a stochastic universal sampling algorithm, that helps to prevent premature convergence by 'holding back' super individuals from taking over the population within a few generations. The best individual is also always allowed to survive to the next generation. Mutation allows for a bit in a chromosome to be changed (mutated). However this operation is given a low probability so as not to change the population randomly. Reproduction is effected by the schema chosen. The crossover point is influenced by how the representation of the rule is expressed. Lankhorst choses a two-point crossover system, thus allowing right-hand side rules to crossover more easily.

The grammars that Lankhorst is learning with the GA are aimed to classify positive and negative examples over a language correctly. Thus the fitness function that is considered takes into consideration the correct classification of positive and negative examples. For various sets of languages, such as 'brackets' up to '0*1*0*1*', the fitness function allows the population to converge into a solution within reasonable results. However, for a micro–NL language, the fitness function did not result in a correct grammar. A further adjustment to the fitness function also allows correct classifications of substrings to influence the fitness score. This modified fitness function allows the GA for a micro–NL grammar to converge correctly.

This work provides an interesting insight to how the fitness of an individual should be calculated. It is obvious that there has to be a certain number of trial and error functions, that will allow us to analyze in which way the fitness effects the convergence of a population. Similarly, in our case we will also want to take into consideration which measures we would like to give more importance to (recall, precision or F-measure). The representation of an individual is also important, Lankhorst having selected a bit–representation over a higher-level representation such as trees. Tree representations can be easily applied with Genetic Programming, and yet, exactly the same principles/discussions apply with respect to the fitness function.

Losee [Los96] applies a GA to learn the syntactic rules and tags with a direct application to document retrieval. The system parses just over 100 abstracts, all about the same general topic, subdivided into 5 different subtopics. The task of the system is to retrieve the documents in the ideal ranking order according to the search term used. The GA is applied to learn syntactic rules and tags, and thus providing linguistic meaning to both documents and terms. As a fitness function, the GA uses a weighted function of the resulting ranking of the document and the average maximum parse length.

Belz and Eskikaya [BE98] attempt grammatical induction from positive data sets in the field of phonotactics. The grammar is represented as finite state automata, and looks at the use of genetic algorithms for this task. In this paper two results are produced, one for German syllables and the other for Russian bisyllabic words. The GA is described in detail, including the type of methods selected, and chromosome representation. The GA used is a fine-grained one, where individuals are on a 2–D grid of fixed size and are only allowed to mate with one of their neighbours (this implementation is referred to as a torus).

An important discussion within this work is the type of representation that should be used for the individuals, which can be either production rules of the form $s_1 \rightarrow as_2$ and $s_1 \rightarrow a$ (where s is a non–terminal symbol and a is a terminal symbol), or a state transition matrix. They argue that production rules produce more fine grained genotype representations since the terminals and non–terminals can be represented individually. On the other hand, state transition matrices can be only seen as a whole, each represented by a single cell. The final representation chosen for this work is that of transition matrices. Each individual represents a possible transition matrix which in turn represents the grammar being induced. In order for genetic algorithms to be used, the matrix is 'flattened' into one string (one row after the other).

The chosen representation has direct implications on the rest of the GA operations. Crossover and Mutation cannot be carried out in the traditional sense of GAs, and certain knowledge must be present in the GA so as to maintain a sound structure of this flat matrix. Belz and Eskikaya seem not to have considered genetic programming, and tree representation for their problem. Such consideration could have provided an interesting outcome with the two applications being compared, and thus being able to decide on which representation scheme would work best in the area of phonotactics.

Keller and Lutz [KL97] work towards learning Context Free Grammars through the use of GAs, by learning probabilities to all possible grammar rules. The initial population of the GA is made of all possible combinations of terminals and non–terminals of the form $A \rightarrow BC$ and $A \rightarrow a$, where A,B and C are non–terminals and a is a terminal. This guarantees that although the grammar is a large one, it is finite. There is also no loss of generality, as all possible rules are present in the initial grammar (including those that will not be part of the final solution).

The type of GA chosen for this work also uses the 2–D grid, in torus formation, where mating occurs only with immediate neighbours. Each individual is encoded as a set of weights, each weight relevant to one parameter. The weight is represented as an n–bit block, and an individual can be viewed as consisting of M blocks of n–bits, where M is the number of all possible rules. Since the grammar is considerably larger than the final solution, Keller and Lutz try to give more importance to 0–probability assignment. Thus, the initial bit/s of the n–bit block is seen as a "binary–switch" as to whether the rest of the bits should be taken into consideration or not.

As for crossover, Keller and Lutz achieved better performance by using a novel genetic operator which they refer to as the *and–or crossover* than by the classical crossover operation. The and–or crossover looks at the parents bit by bit, with one child taking the bits produced by the and operator (conservative), and the other child takes the bits produced by the or operator (liberal). In our proposed work, this idea of the bit-by-bit crossover can be adapted slightly to use functions such as minimum, average and maximum to develop new children.

Spasić et *al.* [SNA04] aim at classifying biomedical terms into specific classes, which represent concepts from an ontology in the biomedicine domain. In order to derive the possible class of a term, they look at the surrounding context of that term. This context is learned through data mining, extracting the contextual patterns surrounding the terms. The patterns contain morph-syntactic and terminological information and are represented as generalised regular expressions. Each contextual pattern is then given a value indicating its statistical relevance. They use this to remove the top and bottom ranked features since they are considered too general or too rare to play a role in term classification. Class selection of a term is then learned using a Genetic Algorithm. For a particular class, the GA tries to learn which of the contextual patterns are relevant. Each individual in the GA is a subset of contextual patterns and its fitness corresponds to the precision and recall of using these patterns on the training data. Eventually the GA learns a good subset of features which can be used to identify terms in that class.

Genetic Programming

Smith and Witten [SW95] propose a GP that adapts a population of hypothesis grammars towards a more effective model of language structure. They discuss grammatical inference using statistical methods, and the problems encountered for this area. They point out that probabilistic n–gram models allow frequent, well–formed expressions to statistically overwhelm infrequent ungrammatical expressions. There is also the problem with allowing probability for unseen data. The 'zero-frequency problem' entails in assigning a small probability to all unseen data, resulting in both ungrammatical n–grams becoming as probable as unseen grammatical ones.

The population is represented as LISP AND–OR S–expressions. Initial experiments showed that certain constraints were required in order for the GP to evolve. These constraints included a maximum depth for nesting and a grammar–generator to allow the GP to evolve towards more suitable grammars. With these constrains in place, the GP evolves simple grammars (e.g. the dog saw a cat) even within 2 generations, however the GP is left to run over more generations to achieve a broader exploration of the search space and hopefully result in a more efficient grammar. Although the results seem positive, there is no comparison to other statistical techniques mentioned that attempt grammatical inference.

Discussion

Representation, particularly for the GP experiment described in Section 3.2, is a central point to all work carried out above. Clearly, there is a lack of knowledge in the area of Genetic Programming, and this is visible in the various attempts of describing a grammar as a flat structure to work within a GA rather than taking advantage of a GP's capability to handle tree representation. Smith and Witten's [SW95] representation of grammars as LISP S–expressions is similar to our idea using extended regular expressions. The resulting grammar that will be produced will be used within *lxtransduce* [Tob05], and any type of representation chosen will have to be translated (either manually or automatically) to the XML format required by lxtransduce.

It is also clear that the fitness function will play a crucial role in determining the success of the experiments. Since we have to our availability manually annotated definitions, precision and recall could be used as part of the fitness function. However, the work described above will be taken into consideration to determine what tests should comprise the fitness evaluation of the population.

5 Conclusions

Attempts at definition extraction have focused mainly on rule-based approaches, with some later work improving results by introducing statistical analysis as a filtering step. Our proposal introduces an element of grammar learning for the set of definitional sentences, influenced by the work carried out in grammatical inference. We will also use the weights produced by the GA as part of the filtering and ranking process, as evidence to what should be classified as a definition. The proposal takes a novel approach in combining GAs and GPs for natural language processing. A quantitative evaluation of these techniques will compare the results achieved to the work carried out so far in definition extraction. The results should be of interest not only to the natural language task of extracting definitions, but also to the machine learning task of combining GAs with GPs.

References

[BE98] Anja Belz and Berkan Eskikaya. A genetic algorithm for finite state automata induction with an application to phonotactics. In *Proceedings of the ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing*, 1998.

[BR07] Claudia Borg and Mike Rosner. Language technologies for an elearning scenario. In *Proceedings of the Computer Science Annual Workshop, CSAW07, Malta*, 2007.

[Fah06] Learning to identify definitions using syntactic features. In *Workshop of Learning Structured Information in Natural Language Applications, EACL, Italy*, 2006.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[KL97] Bill Keller and Rudi Lutz. Learning stochastic context-free grammars from corpora using a genetic algorithm. In *Workshop on Automata Induction Grammatical Inference and Language Acquisition (ICML-97)*, Nashville, Tennessee, 1997.

[KM00] Judith L. Klavans and Smaranda Muresan. Definder: Rule-based methods for the extraction of medical terminology and their associated definitions from on-line text. In *Proceedings of the American Medical Informatics Association (AMIA) Symposium*, 2000.

[Koz92] John R. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[KPP03] Judith L. Klavans, Samuel Popper, and Rebecca Passonneau. Tackling the internet glossary glut: Automatic extraction and evaluation of genus phrases. In *SIGIR'03 Workshop on Semantic Web*, 2003.

[Lan94] Marc M. Lankhorst. Breeding grammars: Grammatical inference with a genetic algorithm. Technical Report CS-R9401, Department of Computer Science, University of Gronigen, PO Box 800, 9700 AV Groningen, The Netherlands, 1994.

[LCN03] Bing Liu, Chee W. Chin, and Hwee T. Ng. Mining topic-specific concepts and definitions on the web. In *Proceedings of the Twelfth International World Wide Web Conference (WWW'03)*, 2003.

[Los96] Robert M. Losee. Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: An empirical basis for grammatical rules. In *Information Processing and Management*, volume 32, 1996.

[MLS07] Paola Monachesi, Lothar Lemnitzer, and Kiril Simov. Language technology for elearning. In *First European Conference on Technology Enhanced Learning*, 2007.

[PBB02] Youngja Park, Roy J Byrd, and Branimir K Boguraev. Automatic glossary extraction: beyond terminology identification. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[SNA04] Irena Spasić, Goran Nenadić, and Sophia Ananiadou. Learning to classify biomedical terms through literature mining and genetic algorithms. In *Intelligent Data Engineering and Automated Learning (IDEAL 2004)*, volume 3177 of *LNCS*, pages 345–351. Springer-Verlag, 2004.

[SW95] Tony C. Smith and Ian H. Witten. A genetic algorithm for the induction of natural language grammars. In *Proceedings IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing, Canada*, pages 17–24, 1995.

[SW06] Angelika Storrer and Sandra Wellinghoff. Automated detection and annotation of term definitions in german text corpora. In *LREC*, 2006.

[Tob05] Richard Tobin. Lxtransduce a replacement for fsgmatch. Technical report, University of Edinburgh, 2005.