Discovering grammar rules for Automatic Extraction of Definitions

Claudia Borg
Department of Computer Science and AI
University of Malta
claudia.borg@um.edu.mt

Abstract

Automatic extraction of definitions from text documents can be very useful in various scenarios, especially in eLearning systems. In this paper, we propose an approach aimed at assisting the discovery of grammar rules which can be used to identify definitions, using Genetic Algorithms and Genetic Programming. By categorising definitions to enable the learning of more specialised grammars, we envisage to improve the performance of our learning programs. A genetic algorithm will be used to learn the relative importance of particular predefined features in definitions. To support this algorithm, we also propose a genetic program to evolve new features from existing ones.

1 Introduction

ELearning is a process of acquiring knowledge through electronic aids, by providing students access to materials that will enable them to learn the tasks. The role of the tutor has shifted from the usual role of teaching in a direct manner to one where he manages a collection of learning materials (visual or textual in digital format) and monitors the students' progress through a Learning Management System.[1]

Documents normally contain various definitions which describe concepts that are required by students as part of the learning process. Thus, automatic definition extraction can be an important component within the elearning environment. Definitions contained within a document will help towards the human conceptual understanding of the texts' meaning, the creation of glossaries and also relate to question answering systems.

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 them 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 this paper we propose an experiment which combines genetic algorithms and genetic programming 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

[1] We will refer to a collection of such materials as a corpus of Learning Objects. However, for the purpose of this work we shall limit the corpus to textual digital documents.

other features to classify the less clear-cut definitions using the features in a combined manner.

The work is carried out in relation to the project LT4eL (Language Technologies for eLearning).[2] The project looks at ways of enhancing the retrieval of learning objects from a LMS by using Language Technologies and Ontologies.

2 Problem Definition

Within a corpus of Learning Objects, texts containing definitions related to the domain of the learning material are normally present. Learning Objects within the LT4eL project have been collected from within the Computer Science and eLearning domains. The objects have been transformed from PDF or Word documents to an XML based format which includes the linguistic annotation with part-of-speech, lemma and morphological analysis information, and retains layout information. Over one thousand keywords and four hundred definitions have been manually identified and annotated for the purpose of keyword and definition extraction tool development and evaluation for each language within the project.

2.1 Defining a Definition

The Oxford English Dictionary defines a definition as a statement of the exact meaning of a word or the nature or scope of something. It normally consists of a *Definiendum*, the term being defined, and the *Definiens*, the expression supplying the definition. In order for a definition to capture a word's meaning, it must describe its operative and functional parts. Context can be used to alleviate ambiguity and reduce the vagueness of a term. Ideal definitions are minimal, integral and talk about the correlation between language and reality. The verb that is used between the definiendum and the definiens is normally referred to as the *connector*, usually providing the relationship between the two (such as 'is a', 'is called', 'means that').

2.2 Towards a Grammar to Identify Definitions

The linguistic information present in the manually annotated definitions is used as a starting point to identify possible grammar patterns that could constitute a definition. Previous work within this area shows that the use of local grammars which match syntactic structures of definitory contexts are most successful in cases where deep syntactic and semantic analysis is not present (Muresan and Klavans, 2002; Liu et al., 2003). The approach throughout the LT4eL consortium was to develop local grammars for the 9 represented languages (English, Dutch, German, Polish, Bulgarian, Maltese, Czech, Romanian, and Portuguese) to extract definition patterns. An XML transducer, *lxtransduce* (Tobin, 2005), is used to match the grammar, which conforms to an XML format specified within the tool, by using XPath. When a match is found, a rewrite rule is applied. In our case we wrap the identified definition in an XML tag. The following is an example of a grammar rule which looks for a determiner at the beginning of a sentence followed by a noun:

```
<rule name="det_S_noun_phrase">
 <seq>
  <query
match="s/*[1][name()='tok'][@ctag='DT']"
  />
  <ref name="noun_group" mult="+"/>
 </seq>
</rule>
```

The initial induction of grammar patterns is being carried out through human observation. Not only is this task a difficult and tedious one, but it is also prone to human error. We have a number of manually generated grammars which capture definitions, but which are of limited use. Przepiórkowski (2007) reports on the difficulties encountered and the results achieved within the LT4eL project for the Slavic group of languages. By trying to discover grammar rules through manual observation, it becomes problematic to decide which features should be generalised and how to tweak rules not to capture non-definitions. The situation is probably aggravated by the attempt of trying to capture definitions in free structured text rather than texts such as encyclopaedias and dictionaries.

The approach taken by LT4eL was to categorise the definitions and develop grammars separately for each category. The Dutch partner within the LT4eL project analysed if improvement over the grammar is possible through the use of deep parsing (Westerhout and Monachesi, 2007), achieving improvement for definitions containing the verb 'to

[2] www.lt4el.eu - An FP6 Project with 12 partners representing 9 languages

be' and other connecting verbs such as 'to mean', 'is called', 'is used' among others.

2.3 Road Ahead

It would be desirable to use a machine learning technique that could automatically learn the relevance of possible sequences of the features described above. The experiment will contribute both to the LT4eL project by producing a novel and automatic approach for definition extraction, and to the scientific community providing further application of machine learning techniques within the natural language processing field and offering comparative results to other techniques in definition extraction. For the purpose of this experiment, we will focus on the English corpus only. However, the work proposed should also be extendable to other languages.

3 Proposed Solution

3.1 From Text to Linguistically Annotated XML

Learning objects are usually created by tutors in different formats such as HTML, PDF or other text formats. In order to standardise a common initial format, all objects were converted to HTML. This choice was based on (i) PDF and all proprietary word processors allow documents to be saved as HTML and, (ii) HTML retains layout information such as bold, italic, headings and so on, which is usually important for conceptual understanding.

Whilst retaining the layout information present in the HTML files, linguistic information is added to each token. Using the Stanford part-of-speech tagger (Toutanove and Manning, 2000) and PCKimmo (Antworth, 1990) for morphological analysis, we are able to add the desired linguistic features to the text and reorganise all the information in an XML based format containing this information as meta-data. A tool to process the HTML files to produce the resulting XML files should view the linguistic tools as plug-ins, so as to ensure the possibility of using other linguistic tools available in the future. It would also be possible to add further meta-data should the need for deeper linguistic processing arise. The XML format also conforms to an appropriate DTD, which is derived from the XCES DTD for linguistically annotated corpora (Ide and Suderman, 2002).

Once the corpus has all the linguistic meta-data available, the manually identified definitions can be annotated and thus become machine-readable. Below is an example of a linguistically annotated sentence which has been manually identified as a definition.

```
<s id="s1501">
<definingText id="dt46" def="m281">
<markedTerm id="m281" dt="y">
<tok id="t20908" rend="b" ctag="NNP"
 base="datum" msd="N,SG,proper,vrbl">
 Metadata</tok>
</markedTerm>
<tok id="t20909" ctag="VBZ" base="be"
 msd="AUX,PRES,S,finite">is</tok>
<tok id="t20910" ctag="VBN" base="define"
 msd="V,PAST,ED,finite">defined</tok>
<tok id="t20911" ctag="IN" base="as"
 msd="CJ">as</tok>
...
</definingText>
</s>
```

3.2 Categorisation of Definitions

In order to simplify the task, definitions have been categorised into six types. This reduces the complexity of the search space, whereby at each grammar identification attempt we focus only on one type of definition. The types of definitions observed in our texts have been classified as follows:

1. Definitions containing the verb "to be" as a connector.
   E.g.: "A joystick is a small lever (as in a car transmission gearshift) used mostly in computer games."

2. Definitions containing other verbs as connectors such as "means", "is defined", "is called".
   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." In this case the term being defined is at the end of the sentence, and it is classified so by the use of "refer to".

3. Definitions containing punctuation features, usually separating the term being defined and the definition itself.
   E.g.: "hardware (the term applied to computers and all the connecting devices like scanners, modems, telephones, and satellites

that are tools for information processing and communicating across the globe).” where the definition is contained within brackets.

4. Definitions containing particular layout style. These can include: a term being defined in point form; the use of tables (similar to the punctuation definition, however the term and definition are in separate cells); the defining term as a heading and the definition is the sentence/s below it.

5. Definitions containing a pronoun, usually referring to the defining term which would be placed outside the definitory context. This is common in cases where the definition is over more than one sentence, and the second sentence would refer to the defining term using a pronoun.
   E.g.: “This (Technology emulation) involves developing techniques for imitating obsolete systems on future generations of computers.”

6. Other definitions to capture those which do not fall in the above categories.
   E.g.: “information skills, i.e. their ability to collect and process the appropriate information properly in order to reach a preset goal.” where the defining term and the definition are separated by “i.e.”

The above classification allows us to be able to generalise rules to identify definitions in categories 1 – 5. However, the sixth categorisation does not facilitate the task of identifying a grammar for this category since it contains exceptional cases, and thus cannot be generalised.

3.3 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 (Holland, 1975; Goldberg, 1989) 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 in Section 3.2, 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 ($1 \leq i \leq populationsize$) will have the following structure:

$$g_i = (\alpha_{i,1}, \alpha_{i,2}, \ldots \alpha_{i,n})$$

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:

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

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.

3.4 Experiment two: Genetic Programming

Genetic Programming (Koza, 1992) 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 learnt 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 emphasise 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 another 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 learnt will be restricted to one category at a time.

3.5 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 learnt 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 learnt by the GA based on the features we manually identified and others that the GP may have learnt. 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 Related Work

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., (2001) built a system whereby glossary candidates are identified automatically, ranked and presented to a human expert, who decides whether they should be included within their system. Rules describing the structure of a glossary item are used in their tool and are primarily made up of parts-of-speech. However, further linguistic tools are placed in a pipeline architecture to fine tune the results based on new linguistic knowledge at every step. Their work was applied to technical texts in the automotive engineering and computer help desk domains. In this type of corpora, glosses are usually well-structured and thus easier to identify using static rules than in a more generic domain. We envisage that the pos structures identified in this work may provide us with features to use in our experiments.

DEFINDER (Klavans and Muresan, 2000) works towards the automatic extraction of definitions in well-structured medical corpora, where 60% of the definitions are introduced by a set of limited text markers (such as ‘-’, ‘()’). Further work (Klavans et al., 2003) looks at the Internet as a corpus, focusing mainly on large government websites, trying to identify definitions by genus. In this task, several problems are identified, both in format and in content. Definitions can be ambiguous, uncertain or incomplete, which have also been encountered in our project. Another problem encountered is that the Internet is a dynamic corpus, and different websites could change their information over time. It also describes decision rules for particular cases, which could be applied to the features in our experiments. 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.

Fahmi and Bouma (2006) 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. A difference between our work and theirs is that we primarily will be using our learning algorithm to learn more generic definitions. However, we plan to run our algorithm also on well-structured, technical texts in order to see the effect of structure in the corpora on the quality of results and also in order to be in a position to compare results to those presented by Fahmi and Bouma.

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 form data. A variety of applications can be identified including syntactic pattern recognition, computational biology, natural language acquisition, data mining and knowledge discovery. Genetic algorithms and genetic programming are two related machine learning techniques which have been applied to grammatical inference.

Work carried out by Smith and Witten (1995) describes a GA that adapts a population of hypothesis grammars towards a more effective model of language structure, with emphasis on inferring practical natural language grammars. The lexical categories are also discovered through the learning process. Of particular interest are the discussions of how constructs such as AND, OR and NOT are used in the representation of the individuals, and how the fitness function, crossover and mutation are selected and applied. These type of constructs are similar in function as to what we propose in Section 3.4 as part of the extended regular expressions.

Similarly, Lankhorst (1994) describes the inference of Context-free grammars through the use of a Genetic Algorithm. The GA is tested to learn languages varying complexity, starting from a rather trivial grammar (equal brackets) to a micro-NL grammar. The former grammar was successfully learnt by the GA, however, the latter was learnt only after the fitness function was changed. This reinforces the importance of such decisions, and the fitness functions described will be taken into account. Loose (1994) attempts to learn syntactic rules and tags with a GA. Again, many components of this work are of interest to us and will be taken into consideration (fitness function, crossover, mutation). However, the most interesting comment is a concluding remark, in that "if one is willing to assume that parts-of-speech are known accurately, the learning of syntactic rules can occur at a much higher rate...". This is encouraging towards the possible achievement of positive results in the experiments described above.

5 Conclusions

We have described the motivations behind definition extraction, difficulties encountered and proposed a possible solution in the form of an experiment. To our knowledge, the use of Genetic Algorithms and Genetic Programming has not been used in the way being proposed for this experiment. 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

Antworth, E. L. 1990. *PC-KIMMO: a two-level processor for morphological analysis.* Occasional Publications in Academic Computing No. 16. Dallas, TX: Summer Institute of Linguistics. ISBN 0-88312-639-7, 273 pages, paperbound.

Fahmi, I. and Bouma, G. 2006. *Learning to identify definitions using syntactic features.* In R. Basili and A. Moschitti (eds), Proceedings of the EACL workshop on Learning Structured Information in Natural Language Applications.

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

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

Ide, N. and Suderman, K. 2002. *XML Corpus Encoding Standard Document XCES 0.2.* Department of Computer Science, Vassar College, and Equipe Langue et Dialogue, Vandoeuvre-lès-Nancy. `http://www.cs.vassar.edu/XCES/`

Klavans, J. L. and Muresan, S. 2000. *DEFINDER: Rule-Based Methods for the Extraction of Medical Terminology and their Associated Definitions from On-line Text.* in Proceedings of the AMIA Symposium. Los Angeles, California, p. 1049.

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

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

Lankhorst, M. M. 1994. *Breeding Grammars: Grammatical Inference with a Genetic Algorithm.* Comp. Science Report CS-R9401, C.S. Department, Univ. of Gronigen, The Netherlands.

Liu, B., Chin, C. W., and Ng, H. T. 2003. *Mining Topic-Specific Concepts and Definitions on the Web*. Proceedings of the Twelfth International World Wide Web Conference (WWW'03)

Losee, R. M. 1996. *Learning Syntactic Rules and Tags with Genetic Algorithms for Information Retrieval and Filtering: An Empirical Basis for Grammatical Rules.* In Information Processing and Management, Vol. 32.

Muresan, S., and Klavans, J. 2002. *A Method for Automatically Building and Evaluating Dictionary Resources.* Proceedings of LREC 2002.

Park, Y., Byrd, R. J. and Boguraev, B. K. 2002. *Automatic glossary extraction: beyond terminology identification.* Proceedings of the 19th international conference on Computational linguistics.

Przepiórkowski, A., Degórski, L., Spousta, M., Simov, K., Osenova, P., Lemnitzer, L., Kubon, V. and Wójtowicz, B. 2007. *Towards the automatic extraction of definitions in Slavic.* To appear in the proceedings of the BSNLP workshop at ACL 2007.

Smith, T. and Witten, I. 1995. *A genetic algorithm for the induction of natural language grammars.* Proc IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing, 17-24, Montreal, Canada.

Tobin, R. 2005. Lxtransduce A replacement for fsgmatch. `http://www.ltg.ed.ac.uk/~richard/ltxml2/lxtransduce-manual.html`

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

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