Rule-based learning algorithm for fact extraction

Peter Siniakov
Project FEx, AG Databases and Information Systems
Computer Science Department, Freie Universität Berlin

Introduction

The area of natural language processing (NLP) gained a lot of attention at the end of 1970es significantly advancing in many research directions such as speech recognition, text understanding, building grammars and conceptual models for natural language. Too optimistic expectations resulting from fast success were not fulfilled and the main goal – understanding and communication in natural language – still remains out of the scope of modern research. In the text-based NLP the focus is being more and more relocated towards solving less complex problems to accomplish very useful tasks in text processing and analysis. One of the most promising efforts in this area is *Information Extraction* (IE).

Most of the information stored in digital form is hidden in natural language texts. Extracting and storing it in a formal representation (e.g. in form of relations in databases) allows efficient querying and easy administration of the extracted data. Moreover, information stored and queried in a canonical way can be processed and interpreted by computers without human interaction; it can serve for establishing ontologies, creation of knowledge bases and data analysis.

The area of IE comprises techniques, algorithms and methods performing two important tasks: finding (identifying) the desired, relevant data and storing it in appropriate form for future use. The notion of *fact extraction* is often used interchangeably with the notion of IE. The goals of fact extraction, however, are typically more specific and according to them fact extraction can be defined as the transformation of facts expressed in natural language to a given, formal, properly defined target structure. The difference to the classical information extraction task should therefore be underlined where the accent is made mainly on the text processing stage and the target representation is less relevant. Fact extraction can therefore be regarded as a subset of IE extraction focusing on more rigidly structured representation forms.

The rule-based approach was the driving force behind first IE systems and has been still the most widely employed, enhanced and improved method in the area of information extraction. Its principle is in providing human syntactic and semantic knowledge which should be sufficient to handle linguistic diversity in a certain domain. However, it suffers from the fact that rules have to be specified manually, which implies large human effort. Classical rule-based approach may be very appropriate for smaller domains but could hardly be employed in considerably large application domains.

To compensate the insufficiencies of classical rule-based approach human effort should be adequately replaced by alternative methods performed by computer. It can happen either generally abandoning the rule concept and using other proven techniques such as statistical, knowledge-based methods or enhancing rule-based method by learning component. The main goal of my dissertation will be developing an algorithm that learns the extraction rules, improves them so that they can be applied to any text from the specified domain to perform actual fact extraction. The algorithm won't depend on any domain and should be universally applicable. The amount of human supervision and training effort should be reduced as much as possible.

Overview

The goal of the described system is to identify relevant facts and data in natural language texts and to transfer them into a predefined formal target structure. The target structure reflects the aspects of the application domain the user is interested in. The processes of finding the relevant facts and relating them to the corresponding elements of target structure form the notion of fact extraction. The system pursues an ambitious aim to perform the process of fact extraction automatically. Target structure and its describing information will be provided by the user since the user defines the focus of fact extraction. Moreover the user will be involved in training of the rule learning component described below.

The input for the system is unstructured, “raw” natural language texts. Many facts can be distinguished by some syntactic regularities; morphological and additional syntactic information might be very helpful for identification of items of interest. Therefore syntactic parser and morphological analyser are applied to the texts. The syntactic information serves as the basis for the following coreference resolution and named entity recognition. Their results are merged with the output of the parser into a complete XML document that contains combined information for each textual element (e.g. paragraph, sentence, word, token).

The algorithm for fact extraction is based on two assumptions. One of them legitimates the use of linguistic patterns saying that similar facts are expressed in similar way and there is a fix set of expressions for every kind of fact. Another one justifies the learning algorithm described below. It states that from concrete linguistic instances of a fact a more general linguistic expression can be formally induced that reflects the semantics of primary instances without expanding it significantly.

According to these assumptions the system tries to learn and improve rules for extraction. A rule consists of a linguistic pattern on the left-hand side and extraction action on the right-hand side. After the system is trained for a certain application domain the learned set of rules can be applied to any text in the application domain. To learn the set of rules the given target structure is analysed and using the provided describing information fragments of text are identified where relevant facts are probably expressed. Rules for extraction of identified facts are automatically generated. After verification of extraction results by the user rules that produce incorrect extractions are removed. The remaining rules are generalized and the learning process continues iteratively.

In the following the components of the system are described in detail.

Syntax parsing, morphological analysis, coreference resolution, NE recognition

Linguistic knowledge can be successfully utilized for the purpose of fact extraction. Since many facts are distinguished by a certain syntactic structure, morphological features or represented by a named entity, the additional information about the linguistic properties of the text is crucial for the success of extraction. POS-tagging and syntax parsing[1] provide the information about the syntactic properties of words and identify low-level syntactic constituents (e.g. noun, verb, prepositional phrases). Morphological analysis finds the essential parts of composed words and identifies stem and principal from of a word, which is very useful for analysis of word semantics. Named entity recognizer that is developed by Christian Siefkes in the FEx project does not only identify named entities, but provides their type. Coreference resolution will be implemented in FEx project in a later stage.

Output of all preprocessing tools is added to the raw text and integrated in a single XML document, which offers a uniform interface for further processing.

[1] Tree tagger (www.ims.uni-stuttgart.de/projekte/corplex/ TreeTagger/DecisionTreeTagger.html) and SPPC (www.dfki.de/~neumann/pd-smes/pd-smes.html) are used for syntactic parsing, POS tagging and morphological analysis

Pattern Language and Unification

The preprocessing unit described above merges and standardizes the information received from different text analysers (syntactic parser, NE recognizer) into a single XML document. Since the format of this document is predefined and fix, pattern generator and unifier do not depend on the peculiarities of preprocessing tools and do not have to be adapted if these are replaced.

To adequately capture the diversity and complexity of natural language a context-free language for formal specification of linguistic patterns has been designed. It allows specifying lexical expressions, syntactic and semantic properties and patterns for XML input or refining the specifications by interleaving the patterns. Besides, it is possible to reference a pattern by a variable for later reuse or, which is more important, for accessing the matching fragment after the pattern has been unified. The specification of patterns can be refined by constraints. For every pattern a constraint restricting the values of certain attributes, its length or distance to other pattern etc. can be formulated. The language for specification of patterns has the prefix property.

The task of pattern unification consists in finding the correspondent text fragment to a given pattern and matching the single elements, particularly unifying the variables of a pattern with the corresponding text elements, in a natural language text enriched by the information of preprocessing tools. The pattern unifier tries to match single elements of a pattern recursively descending to the nested patterns. The variable bindings generated during the matching process are kept in the temporary environment. A pattern matches when every element of this pattern matches and the constraint for the pattern is fulfilled. If an element of a pattern doesn’t match, the backtracking procedure tries to match the preceding elements with other text fragments and the variable bindings acquired during the unsuccessful match attempt are discarded. Eventually, unifying a pattern results either in matching error or unification of variables occurring in the pattern with corresponding text (more precisely XML) fragments. Prefix property of the pattern language helps to avoid ambiguity while interpreting a pattern. On the other hand, different matches of the same text fragment to a pattern are hardly possible. Ambiguous matches can only be caused by using ‘*’ – pattern that matches everything. Since the elements of patterns are specified by numerous properties and constraints ambiguity is very unlikely. If ambiguous matches occur, first match is chosen.

To test the validity of the assumptions in a certain domain or improve the heuristics for rule generalization it can be useful to work with manually created patterns. Therefore pattern unifier provides a parser that translates patterns written in the pattern language into internal data structures for further processing. The data structures reflect fully the pattern language. Another unit translates the pattern data structures produced for example by pattern generator into specifications in the pattern language. This tool is very useful for validating and controlling purposes.

Generation of Patterns

At this stage candidate text pieces are identified as possible fact instances and the patterns encoding them in the pattern language are generated. The natural language text enriched by the information of preprocessing tools serves as the source for generating patterns. The target structure implicitly defines what facts the system should look for. The task of the pattern generator is to close the gap between formal target structure and heterogeneous representation of facts in natural language. In many cases even for a human it would be difficult to conclude what facts are desired given only the target structure. Thus the meta-description of target structure should assist the system in identifying relevant facts. The

meta-description should ideally contain for each element a lexical definition including its semantic denotation, syntactic category, semantic category and associated terms. Semantic denotation is a word that exactly reflects the essence of the fact attribute it stands for.
At first the text can be scanned for the occurrences of semantic denotation, its synonyms and related words. Synonyms and related words can be obtained from the semantic graph, described below, or a thesaurus. Then the syntactic and semantic categories of found occurrences are compared with the meta-description. Depending on a particular case it can also be checked whether the context words belong to the specified semantic category in order to filter irrelevant facts. E.g. if the fact attribute the system is looking for is a verb, its direct object hast to belong to the same semantic category. The occurrence of associated terms or their synonyms in the context also give a hint of a candidate fact. If the recall rate is too low, that is, if a low percentage of facts could be identified, heuristics with deeper semantic analysis can be used. Occurrences of words with the same semantic category can be found and semantic graph can be used to examine the synonym relationship not on the word, but on phrasal level. Besides, sentential locality will be considered when looking for attributes of a fact. After the main attribute of a fact is found, other fact constituents can be searched in its syntactic environment, e.g. in the same paragraph.

The sentence is regarded as the basic syntactic structure for expressing a primitive fact corresponding to the elements of target structure. Patterns are generated by encoding sentences where the possible fact instances are found in the pattern language. The identification and encoding of immediate context, trigger words and the syntactic structure of the sentence are particularly important. Variables are assigned to the constituents representing candidate facts to enable easy access for extraction.

Generation of Rules

Rules are generated based on the set of candidate patterns generated from the training texts. While generating a pattern the information about the element of a target structure that the pattern is related to is preserved. The text fragment that represents a primitive fact and should be extracted is assigned to a variable. A rule is constructed by linking the pattern on the left-hand side with the extracting action on the right hand. Extracting action comprises extraction of textual content from the text fragment (omitting the XML and other auxiliary data) and storage of the extracted content in the target structure. Initially a rule is created for every candidate pattern. After rules have been automatically generated, every rule is applied and the result of extraction is verified by the user.

The goals of fact extraction and existence of predefined target structure suggest that extracting action is the transfer of identified textual content into the target structure. However, since the facts in a target structure are very often composed from primitive facts (attributes), the extraction of an attribute can depend on other performed extractions. In this case the extracting action can be composed from several interdependent extractions. If several candidate text fragments contain the instance of the same primitive fact the extracting action should contain conditions to specify what fragment should be extracted in a certain case.

Generalization and Correction of Rules

Rules generated from candidate patterns suffer to a large degree from two insufficiencies: In most cases they are able to extract a fact if it is in the context which is similar to that the candidate pattern has been generated from. Another major drawback is the fact that rules produce wrong extractions. Ambiguity and diversity of natural language are the reasons why some identified text fragments don’t represent the desired facts and some relevant fragments are ignored by pattern generator. There is always a certain amount of impreciseness when using synonym relationships, working with polysemous words, syntactically ambiguous sentences etc., which misleads heuristics identifying relevant fragments. To account for these insufficiencies rules have to be corrected. The correction is carried out by improving both patterns and extracting actions.

The rule learning algorithm should terminate when a set of rules is available that cannot be improved and satisfies the coverage and correctness constraints. Coverage constraint is fulfilled, if a certain percentage of all relevant fact instances is correctly extracted (recall, that fact extraction implies also storage in the target structure). This means that the recall rate is above a threshold. The set of rules is regarded as correct, if the percentage of correctly extracted facts from all extractions (precision rate) grows over a user-defined threshold. Until these constraints have not been satisfied, every rule in the current rule set is applied to the training corpus. Every extraction of a primitive fact is evaluated by the user who decides whether the extraction has been correct. If it has not, the user classifies the error assigning an error category to the extraction. User interaction may be not necessary if the training texts are tagged with correct extractions. However, tagging can only be done manually and is more tedious than rather simple decision about the correctness of an extraction.

One rule can produce several extractions in the training corpus. If the ratio of the correct ones is big enough, it is added to the initially empty set of “correct” rules. Rules producing a critical number of incorrect extractions are revised. If the modified rule after revision fulfils the correctness constraint it is added to the set of correct rules, otherwise it is discarded. The correct rules are generalized and the algorithm continues iteratively.

The algorithm tries to improve the rules by generalizing elements of rules causing positive extractions and eliminating elements which are responsible for negative (incorrect) extractions. To decide whether a rule can be corrected, all its extractions are examined. If every extraction has been incorrect, the rule is discarded. Otherwise to correct the left-hand side, sentences with positive and negative extractions are analysed trying to identify common elements distinguishing the sentences with positive extractions from those with the negative ones (e.g. certain syntactic structure, specifying adjectives, certain prepositions etc.). Elements occurring in the positive extractions can be included in the rule, elements from the negative extractions can be explicitly excluded using the negation pattern. Using the error category provided by the human supervisor further sources of errors can be eliminated. For instance, if an error is caused by a polysemous word the correct word sense can be specified using the concept-pattern. Right hand side of a rule can also lead to incorrect extractions. Composed and conditional extracting actions can be corrected by removing the erroneous single extractions from the composition and reconsidering the conditions or choosing another candidate fact if there are several alternatives.

A correct rule can still have low coverage because similar facts expressed in a slightly different way may not be captured by the rule pattern. To improve the recall rate for entire text corpus and to extend the range of expressions a single rule is able to extract facts from, rules are generalized. Generalization in this context means replacement of specific elements of a pattern by more general ones provided they are universally applicable, or generating several patterns with similar properties. Generalization can be achieved by generalizing elements of one rule or by merging two rules from a rule set. To account for syntactic diversity of a natural language, patterns with the same lexical constituents, but different syntactic structure are generated. For this purpose the syntactic domain graph is used. Syntactic graph can serve as a grammar when generating expressions with same semantics but a different syntax. Looking for all possible paths containing the syntactic constituents and generating patterns with the syntactic structure corresponding to a path allows creating patterns with the same semantic content, but different syntax.

Often patterns extracting the same primitive fact can be merged because they have a similar structure. After different elements of such patterns have been identified, the more general element replacing them in the merged pattern can be induced. For example, if two partial patterns contain same elements in the different order, they can be unified by the pattern with the arbitrary order of the elements. If two patterns contain the main verbs that belong to the same concept, the merged pattern would contain a verb phrase with any verb belonging to this concept.

Single patterns can be generalized by relaxing the specification of context of the extracted item. Elements of context, which do not significantly contribute to identification of a fact, are identified by heuristics. These elements are either replaced by a more general element or removed. Since both possibilities may have a positive effect, several candidate rules are generated to be verified in the next step.

The information about the predecessors of a generalized rule (rule or rules it was derived from) is preserved. If in the next iterative step of the algorithm the generalized rule produces less correct extractions than its predecessors, the generalization is undone. That is, the new generalized rule is discarded and predecessors of this rule are added to the rule set. This step prevents overgeneralization of rules, which has extraction of many irrelevant facts as a consequence. The predecessors achieved their maximum degree of generality and are marked by a corresponding flag so that they will not be generalized in the next step. The algorithm for generalization and correction terminates when the rule set has not changed after a new step and the criteria mentioned above are fulfilled.

Construction of semantic and syntactic graphs

Semantic and syntactic graphs are conceived as ancillary constructs for semantic analysis and generalization of patterns. Semantic graph captures lexical dependencies of words on a quite low level. The principal forms of head nouns, main verbs and prepositions represent a node in the semantic graph. Head nouns, main verbs, prepositions that are in syntactic constituents immediately following each other in a sentence are connected by a weighted edge. The more frequently two nodes are neighbours in a sentence, the smaller is the length of the edge between them. Other members of syntactic constituent such as adjectives or adverbs in a noun phrase are stored as the characteristic values of the node they belong to. Semantic graph reflects different semantic dependencies between the words. On the one hand, semantic proximity of words indicated by the closeness in the sentence is captured so that synonym relationships and associate terms can be derived. On the other hand, it identifies the property relationship between the main word of a constituent and its subordinate members. Since a thesaurus and other sources of semantics of a natural language are not always available for a certain domain, semantic graph can be the main source for semantic analysis.

Syntactic graph stores the syntactic structure of sentences on three levels. More precisely, there are three syntactic graphs comprising the syntactic structure of the sentence constituents (verb, noun groups), structure of the sentence on the level of syntactic constituents and sentential structure on the level of main and subordinate clauses. The second graph contains syntactic constituents as nodes. Directed edges correspond to the order of constituents in the sentence. The graph implicitly defines a grammar, which is extended by new examples, so that new syntactically correct expressions can be generated given some syntactic constituents.

Application

The training stage ends when the whole training corpus has been processed and a set of rules for extraction of facts has been determined. The set of rules has been optimized to extract facts in a certain domain given a fix target structure. Facts reflected in the target structure can be automatically extracted from new unknown texts. When a new text is processed, at first it is annotated by the preprocessing unit. Rules from the learned rule set are applied to the annotated text. If the left hand side of a rule matches a text fragment, the action on the right-hand side is carried out. The rule may fire several times as there can be several instances of the same fact. Therefore left hand side of each rule is compared with every fragment of the new text. The extracting actions fill the target structure with data. After processing a text the target structure may be partially filled since the text may not provide the entire information required by the target structure. The main advantage of the described system is, however, that the items of interest are automatically extracted and represented in a structured form for the user who doesn't even know the details of the text. The area of application ranges from summarizing sport news to the analysis of business or scientific reports.

References

(Califf & Mooney 98) Mary E. Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing, pages 6–11,Menlo Park, CA, 1998. AAAI Press.

(Cardie 93) Claire Cardie. A case-based approach to knowledge acquisition for domain-specific sentence analysis. In Proceedings of the Eleventh National Conference on Artificial Intelligence,
pages 798–803. AAAI Press, 1993.

(Collier 96) R. Collier. Automatic template creation for information extraction, an overview. Technical report, University of Sheffield,
1996.

(Freitag 98) Dayne Freitag. Toward general-purpose learning for information extraction. In Christian Boitet and Pete Whitelock, editors, Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics, pages 404–408, San Francisco, California, 1998. Morgan Kaufmann Publishers.

(Fürnkranz 99) Johannes Fürnkranz. Separate-and-conquer rule learning. Artificial Intelligence Review, 13(1):3–54, 1999.

(Nobata & Sekine 99) Chikashi Nobata and Satoshi Sekine. Towards automatic acquisition of patterns for information extraction. In International Conference of Computer Processing of Oriental Languages, 1999.

(Riloff & Jones 99) Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In Proceedings of the Sixteenth National Conference on Artificial Intelligence, pages 1044–1049. The AAAI Press/MIT Press,1999.

(Riloff & Schmelzenbach 98) Ellen Riloff and Mark Schmelzenbach. An empirical approach to conceptual case frame acquisition. In Proceedings of the Sixth Workshop on Very Large Corpora, 1998.

(Soderland 97) Stephen Soderland. Learning text analysis rules for domain-specific natural language processing. PhD thesis, technical report UM-CS-1996-087, University of Massachusetts, Amherst, 1997.

(Soderland 99) Stephen Soderland. Learning information extraction rules for semi-structured and free text. Machine Learning, 34(1–3):233–272, 1999.

(Soderland et al. 95) Stephen Soderland, David Fisher, Jonathan Aseltine, and Wendy Lehnert. CRYSTAL: Inducing a conceptual dictionary. In Chris Mellish, editor, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, pages 1314–1319, San Francisco, 1995. Morgan Kaufmann.

(Sudo et al. 01) Kiyoshi Sudo, Satoshi Sekine and Ralph Grishman. Automatic Pattern Acquisition for Japanese Information Extraction. In of Human Language Technology Conference, 2001.