Prerequisites for Fuzzy Inference on Raw Text Using Semantic Reasoner

Olegs Verhodubs
oleg.verhodub@inbox.lv

Abstract. There are a lot of knowledge in the Web, but there is no technological way to extract them. The bulk of knowledge is embedded in texts, and machine text processing is so inefficient that it is necessary to use Semantic Web technologies [1]. Working with ontologies (part of the Semantic Web) is convenient, but the process of creating ontologies is still more of a manual work than an automatic process. This paper proposes to generate IF..THEN rules from raw texts (from sentences) in the Web, and then perform logical inference based on these rules. Moreover, semantic processing is proposed to be applied to the IF part and the THEN part, and not to the entire raw text, generating an ontology from it. This method of generating rules and logical inference is being implemented in the Keyword Search Engine Enriched by Expert System Features [2], which will allow us to obtain expert assessments from many useful texts in the Web.

Keywords: Rules, Fuzzy inference, Expert Systems, Natural Language Processing, Semantic Reasoner

I. Introduction

For thousands of years, the collective experience of mankind has been accumulated in numerous texts. In the 20th century, with the beginning of the Internet era, many of these texts were placed on the World Wide Web, which made them publicly available. However, one problem having disappeared, gave rise to another. The public availability of many materials made their processing and understanding a difficult undertaking. Semantic technologies came to the rescue, namely the Semantic Web. At first, it seemed that a solution had been found, and now texts would become machine-readable. But over time, it turned out that in this case, having removed one problem, Semantic Web technologies create another. Ontologies play a key role in the Semantic Web. Ontologies are well suited for the role of formalization of the subject area. Many ontologies were developed manually. The problem is that many texts cannot be squeezed into the Procrustean bed of a single ontology. Each such text should generate its own ontology. This is long, labor-intensive and not always high-quality if done automatically. The solution is the targeted use of Semantic Web technologies.

Various texts in the Web can be interesting from two points of view: as sources of factual data and as sources of knowledge. Knowledge is useful as a means of reasoning, which makes it possible to predict. This is achieved by building an expert system or rule-based system. If any text in the Web had a ready-made ontology, then in this case it would be enough to generate IF..THEN rules from the ontology and use them in the reasoner. The problem of generating rules from an ontology was covered in the works. However, in the case when there is no ontology for a given text, it is necessary to generate rules from the raw text, and this is also possible, as has already been shown in the works. The generated rules from the raw text cannot be used in the reasoner without changes. This is where Semantic Web technologies come to the rescue, but unlike ontologies, these technologies are used in a targeted manner. Instead of generating an ontology from raw text and then generating rules from the ontology, it is proposed to generate IF..THEN.. rules from raw text and then generate semantic units from the IF and THEN parts. The generated units can then be used in the semantic reasoner. This results in an expert system

that operates on the basis of raw text. This expert system is called an inference-enriched system. That is, the result will be a system that can use various sources for reasoning, both ontologies and raw text.

This paper is organized into several sections. The next one describes possible ways to generate rules from raw text and delves into one of these ways, showing some typical patterns in the text where a rule can be generated. Then, in the next section, it is shown how one can reason about rules that have been generated from the raw text. Conclusions and directions for future research complete this paper.

II. Patterns for generating rules

Each expert system has a knowledge base that is filled with knowledge. The search system is a special expert system, the knowledge base of which is filled with knowledge obtained from various sources. The OWL ontology, as well as the raw text, are the main sources from which knowledge is generated and recorded in the search system's knowledge base. The generation of knowledge, or rather rules, from the OWL ontology has already been described earlier. In turn, the generation of rules from raw text was described in passing: one of the works described the types of sentences from which rules can be generated, while another provided a report on the operation of a computer program that practically implemented the generation of rules from raw text. However, the mechanism for generating rules was not described in detail. It is time to fill this gap.

There are at least two ways to generate rules from raw text. The first way involves using artificial neural networks. An artificial neural network is given a set of test examples one by one, each of which consists of a sentence and a rule that can be generated from that sentence. As a result, the artificial neural network will be trained, which will provide the ability to feed it sentences not encountered in training and will make it possible to obtain a generated rule. The second way involves manually generating sentence patterns from which rules can be generated. Any sentence is a structure, where the elements of this structure are sentence members, such as a noun, adjective, verb, adverb, etc. Thus, a certain set of sentence members is a rule. A certain set of sentence members and the generated rule is a sentence pattern. It is this second method that was practically tested and presented earlier [3]. A full review of proposal patterns with the ability to generate rules requires separate work, although some of this work has already been done [4]. Here we will give examples of just some of them.

For example, there is such a sentence and a rule generated from it:

TABLE I. The first case of sentence and generated rule.

Sentence	Rule
An apple is a green fruit.	**IF** is green fruit **THEN** apple

Using the Python programming language and the nltk module, we find the parts of speech for each word in the sentence:

('An', 'DT'), ('apple', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('green', 'JJ'), ('fruit', 'NN'), ('.', '.')

Here and below, the abbreviations ‘DT’, ‘NN’, ‘VBZ’ and others mean certain parts of speech in accordance with [5]:

TABLE II. A short list of available abbreviations and their meanings.

Abbreviation	Meaning
NN	noun, singular (cat, tree)
JJ	This NLTK POS Tag is an adjective (large)
VBZ	verb, present tense with 3rd person singular (bases)
NNP	proper noun, singular (sarah)
. . . .	. . . .

Thus, such a structure of the sentence:

'DT', 'NN$_1$', 'VBZ', 'DT', 'JJ', 'NN$_2$'

gives the possibility to generate a rule pattern from this type sentence:

IF *JJ NN$_2$* **THEN** *NN$_1$*

Here both *NN$_1$* and *NN$_2$* are nouns, but these nouns are necessarily different, i.e. $NN_1 \neq NN_2$

The words of the corresponding parts of speech in the IF and THEN parts may differ depending on the words in the original sentence.

The following example of a sentence and the rule generated from it looks like this:

TABLE III. The second case of sentence and generated rule.

Sentence	Rule
Chess was invented by Indians.	**IF** was invented by Indians **THEN** chess

Using the Python programming language and the nltk module, we find the parts of speech for each word in the sentence:

('Chess', 'NNP'), ('was', 'VBD'), ('invented', 'VBN'), ('by', 'IN'), ('Indians', 'NNPS'), ('.', '.')

Thus, such a structure of the sentence:

'NNP', 'VBD', 'VBN', 'IN', 'NNPS'

gives the possibility to generate a rule pattern from this type sentence:

IF *VBD VBN IN NNPS* **THEN** *NNP*

Another example of a sentence and a rule generated from it is:

TABLE IV. The third case of sentence and generated rule.

Sentence	Rule
The book, which is on the table, belongs to John.	**IF** is on the table **THEN** book

Using the Python programming language and the nltk module, we find the parts of speech for each word in the sentence:

('The', 'DT'), ('book', 'NN'), (',', ','), ('which', 'WDT'), ('is', 'VBZ'), ('on', 'IN'), ('the', 'DT'), ('table', 'NN'), (',', ','), ('belongs', 'VBZ'), ('to', 'TO'), ('John', 'NNP'), ('.', '.')

Thus, such a structure of the sentence:

'DT', 'NN$_1$', 'WDT','VBZ', 'IN', 'DT', 'NN$_2$', 'VBZ', 'TO', 'NNP'

gives the possibility to generate a rule pattern from this type sentence:

IF *VBZ IN DT* NN_2 **THEN** NN_1

Here both NN_1 and NN_2 are nouns, but these nouns are necessarily different, i.e. $NN_1 \neq NN_2$

One more example of a sentence and a rule generated from it is the following:

TABLE V. The fourth case of sentence and generated rule.

Sentence	Rule
It is necessary to study in order to get a good job.	**IF** to study **THEN** to get a good job

Using the Python programming language and the nltk module, we find the parts of speech for each word in the sentence:

('It', 'PRP'), ('is', 'VBZ'), ('necessary', 'JJ'), ('to', 'TO'), ('study', 'VB'), ('in', 'IN'), ('order', 'NN'), ('to', 'TO'), ('get', 'VB'), ('a', 'DT'), ('good', 'JJ'), ('job', 'NN'), ('.', '.')

Thus, such a structure of the sentence:

'PRP', 'VBZ', 'JJ', 'TO', 'VB', 'IN', 'NN', 'TO', 'VB', 'DT', 'JJ', 'NN'

gives the possibility to generate a rule pattern from this type sentence:

IF *TO VB* **THEN** *TO VB DT JJ NN*

There are many nuances to using rule patterns. One of these nuances is that one sentence can serve as a material for generating several types of rules. For example, from the last example, you can also generate the following rule:

IF *TO VB* **THEN** *TO VB DT NN*

Now, it is clear that it is possible to generate rules from raw (unprocessed) text. Next, we need to consider how to implement logical inference based on the generated rules from raw text.

III. Reasoning over raw text

The presence of rules allows us to build at least question-answering system. Thus, being generated on the basis of a certain text, we get the opportunity to ask questions and get answers strictly on this text. Having generated rules from several similar or dissimilar texts, it is possible to ask questions and get answers on a wider range of topics.

It is much more interesting to implement reasoning based on rules generated from one or more texts. In this case, we already get a full-fledged expert system. Semantic reasoner can help to implement the task of reasoning based on generated rules. Jena reasoner was chosen among many well-known reasoners as the most effective, functional and known to the author [6]. Jena reasoner has several reasoning modes, but the general purpose rule engine is chosen to realize reasoning [6]. Using this mode, it is sufficient to have rules formatted according to Jena's syntax to perform reasoning. For example, the rule "*IF is green fruit THEN is apple*" can be specified as follows:

```
[ruleIsApple: (?a ex:is ex:fruit) (?a ex:is ex:green) -> (?a ex:is ex:apple)]
```

Real life proves that nothing can be stated with 100% certainty. Everything is true with some degree of accuracy or approximation. For example, an apple is a green fruit, but a pear is a green fruit too. It is even possible that a pear is more rightfully a green fruit than an apple. Therefore, it is proposed to use membership functions. For example, an apple is a green fruit with a membership function values [0..0.6], and a pear is a green fruit with a membership function values [0..0.75].

Jena can easily be adapted for fuzzy reasoning, as shown in one of the previous works [7]. For this purpose, you should use built-in primitives such as *lessThan(?x, ?y)*, *greaterThan(?x, ?y)*, *le(?x, ?y)*, *ge(?x, ?y)* and many others [6]. For example, there is the following rule (the values in brackets are the membership coefficients):

***IF** green (>0.5) **AND** fruit (>0.8) **THEN** apple (**min**[0.5;0.8])*

Such a fuzzy rule can be specified in the syntax of Jena as follows:

```
[ruleIsAppleFuzzy:   (?a ex:is ex:fruit)
                     (?v1 ex:fuzzy ?mf1)
                     ge(?mf1,0.5)
                     (?a ex:is ex:green)
                     (?v2 ex:fuzzy ?mf2)
                     ge(?mf2,0.8)
                     min(?mf1,?mf2,?r)
```

```
strConcat('Apple',' ','vmf=',?r,?z)
->
(?z)]
```

Thus, it is possible to describe fuzzy rules. Fuzzy rules are those in which the premise and conclusion can take not only the values True (1) and False (0), but also intermediate values ranging from 0 to 1. However, beyond the specifics, such as describing the possibility of encoding fuzzy rules, attention should be paid to the general system of fuzzy inference. Several fuzzy inference systems [8] are worth mentioning here, but we will only discuss Mamdani's fuzzy inference system in more detail. Mamdani's fuzzy inference system consists of six steps [9]:

1. determining a set of fuzzy rules,

2. fuzzifying the inputs using the input membership functions,

3. combining the fuzzified inputs according to the fuzzy rules to establish a rule strength,

4. finding the consequence of the rule by combining the rule strength and the output membership function,

5. combining the consequences to get an output distribution, and

6. defuzzifying the output distribution (this step is only if a crisp output (class) is needed).

Among the six steps of this inference system, the second step is of particular interest, as it involves automatically assigning membership function values. Although generating rules from raw text is straightforward (this corresponds to the first step of the inference system), and the remaining steps are quite technical and should not present fundamental difficulties, the second step deserves a more detailed examination. The problem is that not all rules generated from raw text correspond to the type of fuzzy rule:

IF (X is A) **and/or** (Y is B) **THEN** (Z is C)

Or in other words, like this:

IF (X is membership function1) **and/or** (Y is membership function2)

THEN (Z is membership function3)

For example, the fuzzy rule shown above corresponds to the following rule:

***IF** green fruit **THEN** tasty apple,*

or in another form:

***IF** fruit is green **THEN** apple is tasty.*

Here “fruit” and “apple” nouns (“NN” in Python nltk module notation) are linguistic variables, but “green” and “tasty” adjectives (“JJ” in Python nltk module notation) are linguistic values. Next, we need to find all the related adjectives to the nouns "fruit" and "apple." For example, green is a color related to the noun "fruit." Related adjectives to the noun "fruit" are different adjectives, namely: red, light green, green-red, etc. Then we can then calculate membership

functions by knowing which adjective is closer to the fruit-apple and which is further away. The same applies to the noun "apple".

If there is no way to find other adjectives related to the noun "fruit" in this or other available texts, then one would assume that "green" would have a membership function value of, say, 0.9, and other colors would have a membership function value of 0.45.

In the case where there is a fuzzy rule of the form:

IF (X is A) **and/or** (Y is B) **THEN** (Z)

for example,

***IF** fruit is green **THEN** apple*

It is necessary to analyze all rules generated from the raw text and find all rules similar to a given rule, both in their assumptions and in their different conclusions. Then, the membership function can be defined as (1 / n), where n is the number of different conclusions under similar assumptions. Otherwise, you can take the value 0.9.

The rule discussed is the simplest, but its pattern is quite common. This means that it is possible to generate quite a few rules from raw text, even using just this pattern. However, as we have already seen, this is far from the only type of rule, and many others exist. It is possible to reason with these rules, but to perform fuzzy reasoning, principles for assigning membership functions must be developed for each rule pattern.

IV. Conclusion

This paper examined in detail the feasibility of generating rules from raw, unprocessed text from a technical perspective. Several sentence patterns from which rules can be generated were considered. The feasibility of organizing fuzzy reasoning based on rules generated from raw text and a semantic reasoner was also demonstrated. All of this is necessary for implementing a Keyword Search Engine Enriched by Expert System Features that can serve as a universal expert system, even without the use of ontologies.

Several directions for developing the Keyword Search Engine Enriched by Expert System Features, the implementation of which is the primary goal of this study, are assumed. First, we can continue to generate sentence patterns and sentence groups from which rules can be generated. Second, we can experiment with more efficient assignment of membership functions to rule parts. Third, we can also experiment with different fuzzy inference systems to ultimately deploy the most suitable one. Fourth, we can consider using a different reasoner, either an existing one or developing a new reasoner specifically for this task. Another direction is to integrate the full range of rule generation capabilities from various sources—that is, not only from raw text, but also from ontologies [10] [11], databases [12], and so on. All of this will ultimately lead to the main goal of the research: the creation of a universal expert system that can extract all possible knowledge from all available sources and inference based on them.

References

[1] J. Davis, R. Studer, P. Warren, “Semantic Web Technologies Trends and Research in On-tology-based Systems,” John Wiley & Sons Ltd, Chichester, 2006.

[2] O. Verhodubs, Keyword Search Engine Enriched by Expert System Features, 2020.

[3] O. Verhodubs, Experiments of Rule Extraction from Raw Text, 2021.

[4] O. Verhodubs, Rule Mining from Raw Text, 2021.

[5] https://spotintelligence.com/2023/01/24/part-of-speech-pos-tagging-in-nlp-python/ [Accessed: 26.09.2025]

[6] https://jena.apache.org/documentation/inference/ [Accessed: 26.09.2025]

[7] O. Verhodubs O., Adaptation of the Jena framework for fuzzy reasoning in the Semantic Web Expert System, 2014.

[8] L. Amaeva, Systems of Artificial Intelligence, 2012 (in Russian).

[9] https://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/fuzzy004.htm [Accessed: 26.09.2025]

[10] O. Verhodubs, J. Grundspenkis, Evolution of ontology potential for generation of rules, 2012.

[11] O. Verhodubs, Ontology as a source for rule generation, 2014.

[12] O. Verhodubs, Generation of Rules from the Relational Database Structure, 2023.