Domain Ontology Learning using Link Grammar Parser and WordNet

Dmytro Dosyn [1], Yousef Ibrahim Daradkeh [2], Vira Kovalevych [3], Mykhailo Luchkevych[1] and Yaroslav Kis[1]

[1] *Information Systems and Network Department, Institute of Computer Science and Information Technologies, Lviv Polytechnic National University (LPNU), Ukraine*
[2] *Department of Computer Engineering and Networks, College of Engineering at Wadi Addawasir 11991, Prince Sattam Bin Abdulaziz University, KSA*
[3] *Department of the theory of wave processes and optical systems of diagnostics, Karpenko Physico-Mechanical Institute of the National Academy of Sciences of Ukraine*

Abstract

The problem of knowledge discovery that comes down to information pertinence evaluation still stay unsolved because of absence of practical effective methods and means of ontology learning. Despite active development of natural language processing tools, they mostly reach the level of semantics, named entity recognition and sentiment analysis but not pragmatics. On the other hand, ontology is the only instrument, “measuring ruler” to compare and estimate the usefulness of information for some particular user, which knowledge could be represented by such ontology as his hierarchical task network (HTN). The need to build separate HTN ontology for each user puts on the agenda the task of design of the automated ontology learning from text. With aim to solve this task the system of automated and semi-automated ontology learning from text had been developed using Carnegie Mellon Link Grammar Parser and WordNet API. Two approaches to distinguish semantic relations in natural language text (NLT) sentences were adopted: analysis of the sentence constituent trees – for explicit relations recognition and Naïve Bayes supervised learning – for recognition implicit semantic relations which need not only verb phrase but other parts of sentence due to its ambiguity. Developed approach was implemented in the Java desktop application using OWL API and Protégé-OWL API. Experimental results were compared to expert analysis and had shown good recognition reliability.

Keywords[1]

Ontology Learning, Natural Language Processing, Naïve Bayes Classifier, Protégé-OWL API, Knowledge Discovery, Pertinence Estimation, Link Grammar Parser, WordNet

1. INTRODUCTION

Traditionally, computer linguistics (CL) presents a text document as a "bag of words", i.e., the model of a document in terms of CL is the distribution of terms according to the frequency of their occurrence in the text of this document. Content as a logical essence is not discussed in this approach, instead only lexical analysis is performed, so it is difficult to distinguish between important and unimportant documents, look for borrowings from other documents, make short annotations, translate their content into other languages. Historically, the vast majority of information retrieval systems have been built on this approach, as the simplest and most obvious. The corresponding information model of the document is called a vector-spatial model. In this interpretation, the document is searched in the space of vectors of frequency of occurrence of terms and their combinations. Such

[1] MoMLeT+DS 2022: 4th International Workshop on Modern Machine Learning Technologies and Data Science, November, 25-26, 2022, Leiden-Lviv, The Netherlands-Ukraine.

EMAIL: dmytro.h.dosyn@lpnu.ua (D. Dosyn); daradkehy868@gmail.com (Y.I. Daradkeh); virusia333@gmail.com (V. Kovalevych); luchkevychmm@gmail.com (M. Luchkevych); yaroslav.p.kis@lpnu.ua (Y. Kis)
ORCID: 0000-0003-4040-4467 (D. Dosyn); 0000-0002-9209-0626 (Y.I. Daradkeh); 0000-0003-3651-5746 (V. Kovalevych); 0000-0002-2196-252X (M. Luchkevych); 0000-0003-3421-2725 (Y. Kis)

search methods are widely described in special and popular literature (see, for example, [1]), however, their main drawback is the already mentioned approach, in which the model of a text document is exclusively lexical and stochastic. This is due to the fact that such models are easy to implement, and therefore this approach to solving the practical problem of information retrieval is reminiscent of a situation where a drunkard, returning from a party, looking for lost keys under a lantern only because it is better visible. But a simple solution is not always correct. And it is intuitively clear that the metric requires choosing an object commensurate with its dimension as a measure of some phenomenon.

Thus, the model of a text document, as a message of one intellectual agent to another has a certain functional structure, caused by its purpose to convey certain important from the point of view of the author information to its recipient. That is why the message usually consists of two parts – a concluding introductory, in which the author indicates to the addressee known to the latter, in his opinion, the context of basic information, and the main, constructive, in which the author transmits to the recipient the information that in a given context will be potentially demanded by the addressee (See Fig. 1). This means that the recipient will supplement the knowledge base with this new information.

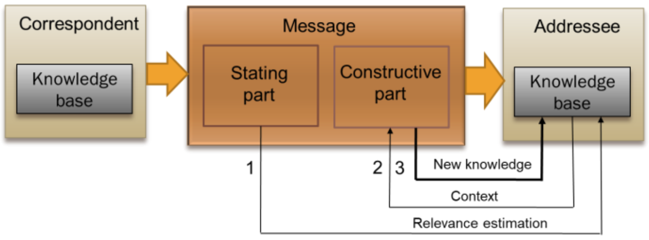


Figure 1: Representation a text document as a message.

The task of the intelligent agent linguistic analysis subsystem is to recognize the true context of the message contained in the text document, provided that it is correctly specified. To simplify, we must assume that this is so. What is the process of context recognition? Obviously, in the localization in the agent's knowledge base of entities and connections, as well as possibly registered in the agent's knowledge base, the facts referred to in this document, ie, to which there is an explicit or implicit reference, but which must be recognized.

To do this, it is necessary to provide the possibility of eliminating ambiguity in the identification of the concepts mentioned in the text. In particular, this is achieved by comparing the attributes of such a concept used in the text with the semantic connections of the corresponding entity that are valid in the agent's ontology. For this purpose it is necessary to perform syntactic-semantic analysis of sentences of this text with their transformation into predicates of descriptive logic, the names of which will be semantic connections between concepts represented in the text mainly by verbs (verb phrases) and adjectives (verb adjectives), and attributes – the concepts themselves, represented by nouns, pronouns or noun groups.

The intention of the author of the message to pass it to the correspondent gives grounds to believe that the author has provided the necessary set of features in the first part of the message to unambiguously identify the context according to the above scheme. After determining the context, the next part of the message can be used to identify in the text of unknown (missing in the knowledge base of the agent) facts and enter them into the knowledge base.

2. THE TASK STATEMENT

The need to build a separate HTN ontology for each user puts on the agenda the task of design of the methods and means of automated ontology learning from text. Those tools should be able to separate the part of input information that can be relevant to the specific user domain, distinguish right semantics of the concepts to be placed at the right position in the ontology's class hierarchy and reconstruct appropriate specific semantic relations between concepts of the learned ontology from a complex verb phase.

Context information should be extracted consequently from the learned ontology in a part relevant to the analyzed text fragment and compared to the discovered context of the whole analyzed document and/or its information source. The most relevant context should be chosen [2].

3. RELATED WORKS

As of 2020, the processing of text documents written in natural language (NLP) is based on statistical models of the use of words and phrases in the text as a whole or the terminological structure of each sentence regardless of the rest of the text, i.e. without reference to the context of the entire message. Thus both the systems of rules developed by means of machine learning of statistical models, and neural network technologies can be used.

The existing ontology population approaches and already available software had been investigated. In particular, similar studies were conducted at Carnegie Mellon University with financial support from DARPA, Google, NSF, and CNPq under the NELL project (Never-Ending Language Learning, [3, 4]). According to the developers, "If successful, this will result in a knowledge base (i.e., a relational database) … that mirrors the content of the Web" (see: http://rtw.ml.cmu.edu/rtw/overview). Initially, the NELL ontology contained several hundred categories of concepts, 280 types of semantic relations between them, and 10 to 15 sample sentences for each of them. The system processed about 500 million Internet pages by selecting possible copies of the categories and connections known to it by several hundred recognition methods available to the system and identified 390 thousand concepts and 350 thousand types of semantic links. Stuart Russell commented the project in his book "Human Compatible: Artificial Intelligence and the Problem of Control": "Unfortunately NELL has confidence in only 3 percent of its beliefs…" and as a conclusion: "Reading requires knowledge and knowledge (largely) comes from reading" [5].

Another similar project, TextRunner, was previously implemented at the University of Washington. During its implementation, about 9 million web pages, 133 million individual sentences were processed, of which 7.8 million tuples were selected [6].

Ontology learning consists in knowledge discovery which differs significantly from data mining and even from knowledge base population. In the case of data mining statistical analysis is used to discover some trends and irregularities in big sets of uniform represented data. The knowledge base population updates the knowledge base by new facts according to already existing formal model of the domain without changing it. Instead ontology learning besides updating separate facts by new data updates and rebuilds whole model.

We define knowledge as useful information. Usefulness anticipates existing an intelligent agent which is able to make decisions taking into account some information which could be (or not be) useful. An agent strategy has almost the same parameter – expected utility. Such coincidence is not accidental – in both cases we are talking about agent's informed decision making. Information that increases agent's strategy expected utility is useful therefore he uses it as knowledge "how to increase his performance". Information usefulness as an agent knowledge called pertinence. Knowledge metrics problem still stay open because of absence of adequate measure instrument which could be ontology itself, but only in the case when it will contain an optimal strategy of the ontology owner. Formal description in an ontology of optimal strategy with states, rewards, actions, goal and evaluated expected utility creates an agent's knowledge what to do to reach max utility (reward) in observed conditions.

How to build an agent optimal strategy by means of RDF-OWL-SWRL-SPARQL-PDDL ontology – still stays an open problem, but some prospective approaches had been already developed [7…9].

Each separate word and any unordered set of words can't represent the same statement as a logical sentence with right syntax, similarly no one set of not ordered sentences (and correspondent DL predicates) can't be distinguished as a valuable new knowledge without asserting them into existing DL model. The key problem of discovering new knowledge in a text document written in natural language is the development of a method of automatic learning of ontology as a formal-logical model of the world, in which new information found in the text can be transformed into knowledge.

According to information theory by Claude Shannon [10] any knowledge as a kind of information could be measured in terms of changing entropy of the system. But it could be shown that not all equal entropy changing gives to an intelligent agent equal increasing his performance or equal opportunity to reach closer to his goal. Exactly the change of expected utility of the agent strategy after updating his knowledge base by this new information could be used as its usefulness i.e. pertinence. Therefore to find if this information is useful for this agent, other words, to estimate the pertinence of the information we should include it into ontology strategy of the agent, rebuild an optimal strategy, taking into account new information, evaluate new expected utility, compare it to previous one and take the difference as a value of pertinence of a new information, i.e. obtained knowledge value. For this purposes an adaptive self-learned strategy planning ontology of the intelligent agent, based on PEAS architecture is needed together with ontology learning technics similar to described here.

In general, the solution of almost any automatic word processing problem includes the following levels of analysis:

• graphematic analysis – to be concise if simplified – the division of the text into individual words (tokens) and sentences;

• phonetic analysis, applicable in the processing of acoustic data;

• morphological analysis – selection of the grammatical basis of the word, definition of parts of speech, reduction of words to the dictionary form;

• parsing – identifying syntactic connections between words in a sentence, building the syntactic structure of the sentence;

• semantic analysis – identification of semantic connections between words and syntactic groups;

• pragmatic analysis – identification of statements – facts and rules, assessment of the importance of the identified statements for the overall software model.

Each such analysis is set and implemented as a separate independent task, which is used in conjunction with others to solve specific applied and theoretical problems.

A. TECHNOLOGIES

From the point of view of practical realization, the most developed and, accordingly, the most widespread are the following methods and corresponding NLP technologies:

1. Named entity recognition (NER) is one of the most popular methods of semantic analysis. According to it, the algorithm takes a phrase or paragraph as input and identifies all existing nouns or names by classifying them. The task of fully recognizing a named entity can be divided into two different problems: identifying names and classifying names according to the type of entity they refer to (e.g., person, organization, location, and others). Hierarchies of the named types of entities are developed. For example, BBN categories [11] consists of 29 types and 64 subtypes, Sekin's extended hierarchy consists of 200 subtypes [12]. Ritter used a hierarchy based on conventional Freebase entity types, applying NER to texts from social media [13]. Approaches with the use of social networks data are considered promising [14].

2. Tokenization or lexical parsing is the process of converting a sequence of symbols into a sequence of tokens (groups of symbols corresponding to certain patterns), and determining their types. The process of displaying texts as a set of characters in a set of ribbons, a set of ribbons in a set of words are the main steps of any PMT processing task. Various methods and libraries are available for tokenization, in particular, such as NLTK, Gensim, Keras.

A tokenizer based on Penn TreeBank [15] allows to recognize punctuation, auxiliary words (words that occur only in combination with other words) and words with a hyphen. Moses' tokenizer developed as part of the project of the automatic translator of the same name (http://www.statmt.org/moses/) based on statistical training on parallel text corpora.

spaCy (https://spacy.io/api/tokenizer) – another tokenizer which provides flexibility in defining special markers.

Word particle tokenization is also used. This tokenization is very useful for a specific application, where it is important to take into account frequently used particles [16].

3. Stemming and lemmatization. In linguistic morphology, stemming is the process of removing affixes in order to obtain the basis of a word, which will serve as its most general formulation. The base found does not have to be identical to its root. Since 1968, in the field of computer science, appropriate stemming algorithms have appeared. Many search engines use stemming to highlight the base in order to supplement the queries with synonyms which is called a conflation.

Lematization is aimed at solving the same problem with the only difference that special dictionaries (for example, WordNet [17]) and/or corpora of texts are used to find the basis in order to find the correct vocabulary word form – a lemma, as some canonical form of the studied token.

4. "Bag of words"/TF-IDF/N-grams model – NLP procedures, aimed at further its lexical and statistical analysis as a vector-space model [9] for using machine learning methods. This procedures preferably includes mentioned stages of tokenization, stemming, and lemmatization. This approach ignoring context but is simple therefore widely popular.

5. Sentiment analysis has been actively developing since the 2000s. Most commercially developed NLP systems include a subsystem to analyze their tonality (see, for example, [19]). In such an analysis, it is assumed that the textual information contains both facts and judgments of the author, the tone of which should be determined. Knowledge-based techniques, statistical methods, and hybrid approaches are used to distinguish the polarity of the studied text messages [20].

6. Immediate constituent analysis (analysis of successive layers – constituents) is a method of sentence analysis, which was first formulated by Leonard Bloomfield, and developed by Rollon Wells and Noam Chomsky [21, 22]. The method uses successive division of a sentence into subparts – “immediate constituents” up to last word to obtain a layered sentence structure of recursive inclusions, which should represent some its internal logic to be further analyzed. Now this practice is widely used.

7. Dependency analysis parses syntactic structure of a sentence to build a network of subordinations dependencies between all separate words of the sentence. Used dependency grammar lets to reveal the basic elements of the sentence, which fulfils the roles of subject, object and a semantic relation between them as skeletal predicate structure of the sentence. Authors [23] used to compare constituent and dependency analysis by parsing the same example for both methods (see Figure 2).

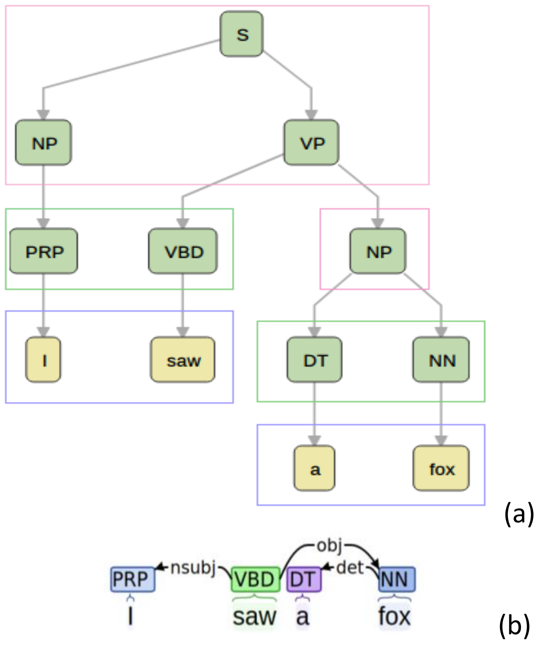


Figure 2: Immediate constituent analysis (a) and dependency analysis (b) of the sentence “I saw a fox” (From https://www.baeldung.com/cs/constituency-vs-dependency-parsing)

The result of dependency analysis can be represented by an oriented graph with a single root node corresponding to the predicate, from which the edges named according to the type of connection are directed to the dependent members of the sentence, and each word in the sentence, except that which serves as a root node, must be at least one edge oriented to that word. Although syntactic dependencies cannot be directly transformed into predictive logic predicates, they, in combination with constituent analysis data that allow the identification of multi-word predicate attributes, can serve as a sufficient basis for the application of machine learning methods to classify-recognize known semantic connections and corresponding predicate constructions.

B. TOOLS

At the end of the second decade of the 21st century, it is the methods of automating the processing of text written in natural language, or rather their imperfection, became a bottleneck in the development of information systems towards their intellectualization: over-scale volumes of available textual information remain unavailable for full analysis. The developed tools use vector-space model methods such as "bag of words" or recognition of named entities and do not go beyond the analysis of the tone of the text, automatic abstracting or translation. A number of companies and research groups are actively developing and improving existing systems and technologies:

Table 1
The most commonly used NLP tools [24]

Name	Prog.	Vers.	Algo	Classifiers
StanfordNLP	Java	4.2.0	CRF	CoNLL, MUC6-7, ACE
OpenNLP	Java	1.9.3	Max. entropy	
SpaCy	Python	3.0.5	Neural (2.0)	OntoNotes
NLTK	Python	3.5	Max entropy	StanfordNER
GATE	Java	9.0.1	JAPE	
Link Parser	Java	5.8.1		Link Grammar

1. **NLTK** (Natural Language Toolkit) – open source library of entry level NLP tools, developed with Python. It includes tools such as:
• tokenization;
• stemming;
• grammar markup;
• text classification;
• NER;
• sentiment analysis.

In addition to the NLP processing tools themselves, NLTK includes more than 50 lexical databases and text corpora, such as RST Treebank [25], Penn Treebank [15], WordNet 17], Lin thesaurus [26] etc.

Drawbacks include low performance and lack of a syntactic parser, which must be compensated by additional tools [27], such as, for example, the statistical parser Bohnet and Nivre [28].

2. **Stanford Core NLP** – multi-purpose “full-stack” tool for text analysis, including NER, text tonality (sentiment analysis), construction of conversational interface. Like NLTK, Stanford CoreNLP offers a wide range of NL text treatment tools. The main advantage of Stanford NLP tools is scalability. Unlike NLTK, Stanford Core NLP is better suited for processing large amounts of data and performing complex operations. Thanks to the architecture of the scalability system Stanford CoreNLP effectively handles the processing of streaming data, such as extracting information from open sources (social media, breaking news, running reviews generated by users), tone analysis (social media, customer support), conversational interfaces chatbots), natural language generation (customer support, e-commerce). This tool effectively recognizes named entities and provides fast syntactic markup of terms and phrases.

3. **Apache OpenNLP** – a multifunctional text analyzer similar to Stanford Core NLP, which supports the main common tasks of natural language text processing, such as tokenization, sentence segmentation, tagging of parts of speech, recognition of named entities, fragmentation, parsing,

language recognition and interpretation of current references (searches expressions that refer to the same entity in the text). This analyzer, according to its documentation, is based on the use of machine learning methods, such as classification by maximum entropy and perceptron modeling. Apache OpenNLP is a library that consists of many components. The user can make from these components a NLP conveyor according to his tasks.

A comparative analysis of the Apache OpenNLP on the Internet with Stanford Core NLP shows a significant functional similarity, which does not allow making an unambiguous decisive choice between them: both systems perform their functions equally well.

4. **SpaCy** – open source software library for NLP in Python. The library includes the following components:

• tokenization
• POS tagging
• dependency parsing
• lemmatization
• SBD (sentence boundary detection)
• NER (named entity recognition)
• EL (entity linking)
• similarity
• text classification
• rule-based matching – search for patterns by given features and regular expressions.

In general, this is a standard set of basic-level functions for the construction of specialized PMT processing systems designed to solve specific problems in particular software. The main feature of the library, which distinguishes it from the previous two, is the implementation in Python. Easy integration of deep learning software is declared. It also includes proprietary parsing and NER visualization tools.

5. **GATE** – open-source software infrastructure and graphical IDE for reliable creation and deployment of components and resources for NLP [29]. Due to the modular architecture, it can be used element by element or complex. It performs an almost complete set of the above operations with text. Initially it was made in Java but currently also supports Python. The low entry threshold for system users is provided by the existing graphical shell, which ensures its widespread use not only by experts in the field of NLP, but also by those who try to solve practical problems in other areas related to streaming and processing text streams, such as automatic dialog systems.

The advantages of GATE developers include the following properties of the system:

• automatic measurement of the performance of components;
• distinguishing between low-level tasks, such as data storage, data visualization, component detection and loading, and high-level NLP tasks;
• clear separation between data structures and NLP algorithms;
• use of standard linguistic data exchange mechanisms for components and open standards such as Unicode and XML;
• correction of exotic data formats – GATE performs automatic format conversion and provides alignment and standardization of NLP data;
• providing a basic set of NLP components that can be expanded and/or replaced by users as needed [30].

An important positive feature of the GATE project is the presence of APIs for modeling ontologies and manipulating them.

6. **Link Grammar Parser** – a publicly available library of NLP software, originally built on ANSI C as a formal grammar system Link Grammar, which in the process of development became part of the Java distribution RelEx (https://wiki.opencog.org/w/RelEx_ Dependency_Relationship_Extractor) [31]. The system was developed in the 1990s by three linguists – David Temperley, John Lafferty and Daniel Sleator [32]. This system includes all the elements of text analysis – from the initial (graphematic) to the level that can be called the primary semantics of the English language.

The system decomposes English sentences into a set of binary meta-semantic relations between words or punctuation, thus forming some syntactic structure (similar to dependency parsing). To do this, the authors have built a special dictionary, in which for each word of the language indicates which meta-semantic relations it can be associated with other words in the sentences of this language.

Link Grammar Parser (LGP) distinguishes over 100 different types of meta-semantic relations between words in English sentences, providing the user with the ability to recognize detailed context.

The advantages of the system include the ability to process the entire array of text in parallel, provided by dictionary technology to identify meta-semantic relations between word pairs in sentences with subsequent weight filtering by probability, which is fully consistent with the Big Data view. At same time due to the exponential complexity of used algorithm it operation is limited in time.

4. METHODS

Among all the above-mentioned NLP tools and related technologies, it is necessary to choose the most appropriate for the task of knowledge discovery according to the criteria of reliability, robustness, speed and ease of implementation.

From the point of view of algorithmic language, the choice was made in favor of Java, mainly given that most software libraries for work in the field of semantic networks (in particular, Protégé-OWL API, OWL API, Apache Jena) were developed in this language. From the point of view of specific distributions and relevant software libraries, it is quite difficult to choose one (see Table 2) and, based on previous development experience, it is necessary to avoid such a choice as long as possible, trying to choose the most standardized solutions and combine necessary elements of various means (API) by including the required software libraries in the overall project. At the same time it is necessary to pay special attention to possible conflicts between versions and implementations of the involved libraries.

Table 2
Some aspects of implementing NLP tools

Software	URL	Lang.	Licence	OWL API
NLTK	https://www.nltk.org/	Python	Apache License 2.0	-
Stanford Core NLP	https://nlp.stanford.edu/	Java	GNU GPL	+
Apache OpenNLP	https://opennlp.apache.org/	Java	Apache License 2.0	+
SpaCy	https://spacy.io/	Python	MIT License	-
GATE	http://gate.ac.uk/	Java	LGPL	+
LGP (RelEx)	https://github.com/opencog/relex	Java	Apache License 2.0	+

Among other dependency parsers, RelEx attempts to obtain a greater degree of semantic normalization: for questions, comparative data, entities, and sentence and subordinate clauses, while other parsers (such as Stanford Core NLP) follow a literal representation of the text's syntactic structure. In particular, RelEx pays special attention to determining when a sentence is hypothetical or speculative, and isolating query variables from the question. Both of these aspects aim to make RelEx suitable for question-answering and semantic understanding/reasoning systems. In addition, RelEx performs word markup to denote a part of speech, a set of nouns, their gender, the temporal form of verbs, and so on. According to the developers, RelEx analyzes text almost four times faster than Stanford Core NLP, and it provides a "compatibility mode" in which it can parse sentences in the same format as Stanford University's parser.

Nevertheless, RelEx project seems to be already stopped, moreover, none of the mentioned systems contains the means of analysis at the level of pragmatics of text documents. The vast majority of them are about equally successful in analyzing the tone of the text. Heuristics based on statistical criteria using machine learning methods are mainly used. And here there is nothing unusual because text message pragmatics (its assessment in terms of usefulness-neutrality-harmfulness) cannot be determined without the use of knowledge base formulated as an optimal strategy of the intellectual agent, as there is no starting point – the person with his goals and/or damages to be assessed.

Thus, it was decided to combine different libraries in the project as much as possible, preferring the most standardized ones and those that do not impose unjustified restrictions on possible further selection and development. As a core of the software package stays the OWL API library which should help to reach the main aim – populate OWL ontology as a knowledge keeper and simultaneously tool of text pragmatics evaluation.

A. USED TECHNOLOGIES

Despite the need to combine different NLP technologies due to the lack of full-fledged means of extracting from the text description logic predicates necessary for ontology learning and/or knowledge base population, one of them had to be taken as basic and supplemented by missing algorithms and technologies. For the reasons stated above, the choice was made on the approach used in the Link Grammar Parser. The main advantage of the method is simplicity and usability. Parsing mechanism is separated from dictionaries describing language. Due to the use of dictionaries and the system of rules of syntactic dependencies between words of the sentence, which can be dynamically edited and supplemented in the learning process, this parser easily adapts to a given problem area, organically combines with the ontology, actually becoming part of it. On the other hand, as mentioned earlier in the parser's overview, the ability to recognize more than 100 types of meta-semantic relations between words allows for detailed semantic analysis of the text.

Link Grammar Parser focuses on parsing syntactic dependencies between sentence words. However, there are a number of significant differences: the edges of the formed relations graph are not oriented. The parser also analyzes the dependencies between phrases, thus combining the advantages of both approaches – constituent analysis and dependency parsing. As a result, the recognized relations between words and phrases are not only syntactically but also contextually dependent. Therefore, the developers have used their own system of notation for such dependencies, which would, in their opinion, best suit their role in shaping the semantics of a sentence formed from related words and phrases. In this regard, it is appropriate to call the syntactic dependencies classified by the developers of LGP as meta-semantic relations (MSR).

As combined identifiers of such meta-semantic relations, special symbolic sequences are used – connectors that correspond to a specific type of connection. An either '+' or '-' sign is attached to each connector on the right to indicate the direction of relation. For example, if the word W1 is assigned the connector A+ and the word W2 is assigned the connector A-, then in the syntactic structure of a sentence consisting of two words W1 and W2, the connection A will be established between the words W1 and W2. In this case, the sentence "W2 W1" will not receive any interpretation, because W2 is assigned MSR A-, which forms a connection only to the left, and the word W1 is assigned A+, which forms a connection only to the right.

LinkParser performs the analysis in two stages: the first stage – construction of set of syntactic representations of one sentence in English. The parser considers all variants of meta-semantic relations between words, choosing among them those which satisfy criteria of projectivity (connections should not intersect) and connectivity (the resulting graph should contain as few components as possible).

For example, for the following dictionary:
the: D+
white: A+
horse: D- & A- & S+
was: S- & PP+
tired: PP-
the phrase "The white horse was tired" will be parsed as follows:

```
+------D-----+
|     +---A--+-S--+-PP-+
|     |      |    |    |
The white horse was tired
```

Here, the algorithm distinguishes four pairs of words connected by the corresponding MSR: D (the - horse), A (white - horse)), S (horse - was), PP (was - tired).

In the second stage of analysis, the program divides the constructed graph into so-called domains, each of which contains links belonging to the corresponding fragment of the sentence (atomic statement). For example, in the sentence You think that I am happy, two domains "You think" and "that I am happy" will be selected. Then for each domain conditions of type are checked: if in the domain there is ‘A’ relation there should (or should not) be a relation ‘B’. Thus, it is possible to check presence/absence in the domain of one relation depending on presence/absence of other relation.

A few MSR can be attributed to one word. Their combination may be subject to certain conditions, reflected in the dictionary by notation similar to that used in regular expressions, namely:

& – "asymmetric conjunction": "W: A+ & B+" means some word X, with which the word W forms a connection A, must be earlier in the text than the word Y, with which the word W forms a connection language B;

or – "disjunction": "W: A+ or B-" means the word W can form either a connection A to the right or a connection B to the left;

{} – "optional": "W: A+ & {B+}" means after the word W formed the right connection A, it may or may not form the connection B;

@ – "unlimited" means that the connection can be formed an unlimited number of times.

Formulas can be assigned to a single word or a whole class of words. For example, the connector (A) that connects adjectives with nouns is assigned to all adjectives at once. Similarly, the formulas are assigned to 23 categories of English words, distinguished by morphological (degrees of comparison of adjectives and adverbs, grammatical number (singular-plural) nouns, etc.) and basic syntactic features (transitivity, bi-transitivity, etc.).

Traditional relation extraction approach uses extraction templates to distinguish predefined set of relations, training documents and testing documents as well [33]. Here it is proposed to build extraction templates using patterns, automatically constructed from set of most used word-MSR-word pairs. LGP parser gives us all needed preliminary information to build such patterns.

The choice of a particular method took into account the features of the problem facing the system as a whole, the initial stage of learning, as well as the architecture of the system, namely the presence of an ontology as both a means and objectives of such learning:

1) terminological TBox facts should contain statistical templates of semantic connections, by which the system will recognize a potential binary predicate;

2) search-recognition-learning should be purposeful, in the direction of solving some specific problem (set of problems) given by the ontology of the subject area;

3) the applied algorithms should have the minimum complexity because of need of their repeated in a long cycle of check of expected usefulness of the received fact.

The processing of NLT in this way should consist in the sequential transformation of sentences of the text into predicates of descriptive logic in terms of a pre-formed (learned in the same way) ontology.

B. ALGORITHM

The general flowchart of data transformations for extracting semantic relations from text is presented on the Figure 3, corresponding block-diagram of the algorithm is presented on the Figure 4.

Software implementation of the method of ontology learning involves recognizing semantic relations in NLT sentences, building on their basis DL predicates, identifying their logical interdependence in order to form appropriate rules and assert the identified facts and rules into the ontology.

To solve this task it is necessary to take into account the following statements:

1. A prerequisite for identifying a semantic relation is the presence of a verb phrase. Its presence is determined by means of direct constituent analysis, and the type – corresponding to the verb phrase meta-semantic relation. All conjunctions to the right of the conjunction 'S *' indicate a verb phrase.

2. The words that accompany (which are pointed to by) these symbolic sequences of meta-semantic relations determine the type of semantic relation, i.e, the name of the predicate encoded in the sentence. These words are usually verbs: "knows", "has", "belongs", "refers", or verb phrases "belongs to", "consists of", etc.

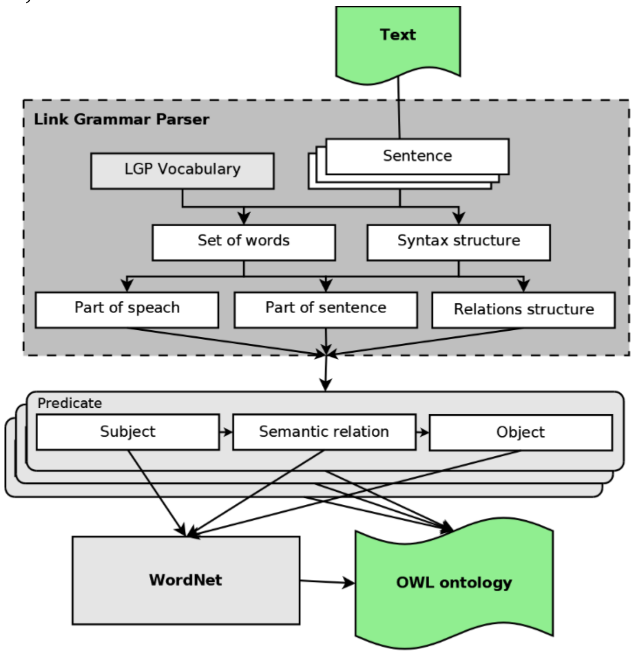


Figure 3: General flowchart of data transformation for a separate relation extraction using Link Grammar parser and WordNet.

3. Adjectives are interpreted as properties and can also be recognized in sentences through the semantic relation "has a property".

4. To recognize the semantic relation, the following steps should be performed:

- parse sentences using LGP;
- find a verb phrase through constituent parsing taking into account the connection symbols to the right of "Ss" MSR;
- find the verbs indicated by these symbols;
- find the subject of the action (subject in the sentence), which indicates the symbol "Ss";
- find the object of action (obviously, the definition in the sentence), i.e., the subject to which the action is directed;
- check in the ontology the presence of this type of semantic connection and, in its absence, create it;
- check the presence in the ontology of entities that mean the object and subject of action. There are different options (see Table 3 below).

Table 3

Options for ontology learning under different conditions of presence in the ontology of components of the detected predicate

#	Existing in the ontology			Possible action
	Relation	Subject	Object	
1	+	+	+	Assert in ontology
2	+	+	–	Add an unknown concept to the ontology
3	+	–	+	Add an unknown concept to the ontology
4	+	–	–	Ignore
5	–	–	–	Ignore

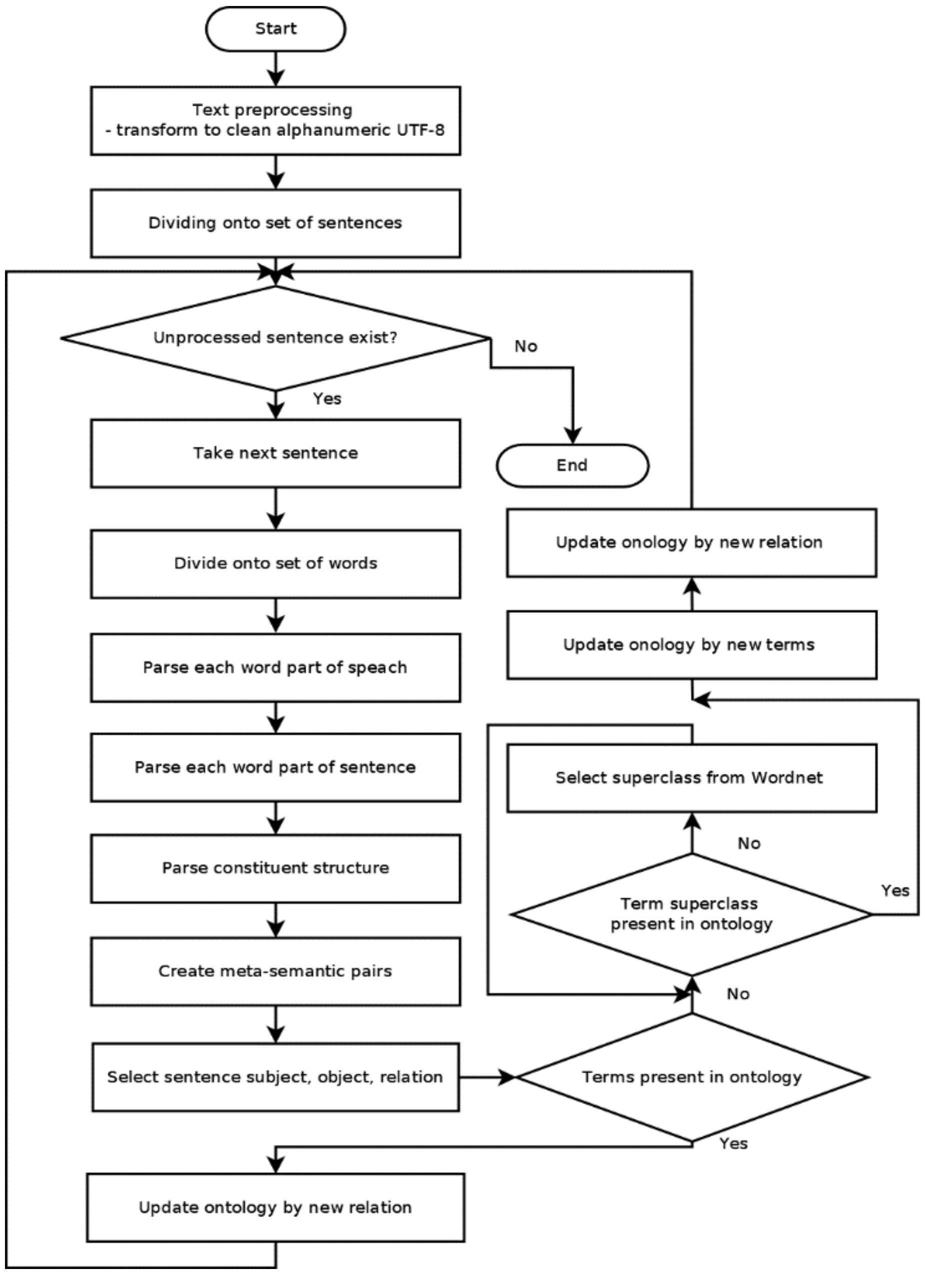

Start
Text preprocessing
- transform to clean alphanumeric UTF-8
Dividing onto set of sentences
Unprocessed sentence exist?
No
Yes
Take next sentence
End
Update onology by new relation
Divide onto set of words
Update onology by new terms
Parse each word part of speach
Parse each word part of sentence
Select superclass from Wordnet
No
Parse constituent structure
Yes
Term superclass present in ontology
Create meta-semantic pairs
No
Select sentence subject, object, relation
Terms present in ontology
Yes
Update ontology by new relation

Figure 4. A block-diagram of the algorithm of ontology updating by a separate DL predicate using Link Grammar parser and WordNet.

In case (5), obviously, the sentence is ignored. In case (4) also nothing to connect: so, for example, if the known ontology connection 'IS-A' is used, which connects two unknown ontologies terms: X and Y, such statement will also have to be ignored. In cases (2) and (3) it is possible to introduce the concept into the ontology through a known semantic relation.

5. A distinction should be made between ontology as a set of admissible semantic relations between concepts and the knowledge base, as a set of facts and rules about a given model of reality.

6. Relations can be unconditional and conditional. Conditional relations are written as rules – implication of predicates. Unconditional connections are a partial case of conditional ones and are recorded as facts in the form of separate predicates.

7. It is necessary that the recognition of semantic relations is based on the data of the learned ontology which, among other things, would contain the concept of semantic relation and its features-properties.

Based on the above arguments, a semi-controlled machine learning method using the naive Bayes classifier was chosen. To do this, we used the LGP analysis of the NLT sentence, which resulted in a graph of meta-semantic relationships between the words.

In the first step, the ontology recognizes the words of this sentence as concepts of a given domain. Next, the results of parsing identify the roles of recognized words in the sentence. The next step is to build a feature vector of each of the recognized concepts in relation to its role in this sentence. This vector of features recognizes the type of semantic relations in a sentence. The ontology statistics is accordingly updated which is one of ontology learning stages. Since the basis of the semantic relation in a sentence is a verb phrase, the analysis begins with it.

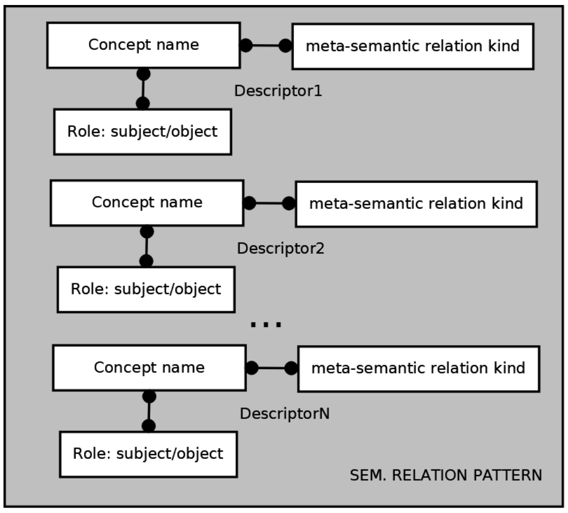


Figure 5. Relation extraction template implemented

When analyzing another sentence, the system recognizes or does not recognize the verb phrase selected by the direct constituent analysis of this sentence. If it does not recognize, it enters it into the ontology as a new type of relation and creates its pattern (see Figure 5) with the ability to calculate statistics. If it recognizes, it adds an existing pattern to the statistics. If in the ontology there are several semantic relations with the given verb, the system by means of procedure of recognition on statistics of patterns decides to what type of relation this sentence (the given verb phase) concerns.

The classical relation was used to recognize the type of semantic relation by the naive Bayes classifier:

$$P(h_i \mid X_j) = \frac{P(X_j \mid h_i) P(h_i)}{P(X_j)} \tag{1}$$

where h_i – i-th relation hypothesis; X_j – feature vector (pattern) of j-th semantic relation; $P(h_i)$ – unconditional probability (frequency of occurrence) of the i-th type of semantic relation; $P(X_j)$ – probability of appearance of the j-th feature vector (frequency of occurrence of the j-th pattern); $P(h_i \mid X_j)$ – probability that the i-th type of semantic relation takes place in the presence of the j-th feature vector; $P(X_j \mid h_i)$ – the probability of the appearance of the j-th feature vector, provided that there is the i-th type of semantic relation.

The naive Bayes classifier is used, for which a feature vector is considered statistically independent, and therefore the probability of co-occurrence of features is considered equal to the product of the probabilities of independent appearance of each of the features:

$$P(X_j \mid h_i) \propto \prod_{k=1}^{N} P(x_{k,j} \mid h_i). \tag{2}$$

Then the probability of belonging to the class h_i, provided that the set of features X_j is close to:

$$P(h_i \mid X_j) \propto \prod_{k=1}^{N} P(x_{k,j} \mid h_i) P(h_i), \tag{3}$$

that is, we are looking for a hypothesis about the present semantic relation h^*:

$$h^* = \arg\max_i \prod_{k=1}^{N} P(x_{k,j} \mid h_i) P(h_i), \tag{4}$$

where N – the number of features of the pattern.

This method was implemented as a Java software using the Protégé-OWL API and Link Parser API as part of the overall CROCUS project. After preprocessing the text document is divided into separate sentences and for each of the sentences it is sequentially parsed by LGP, the results of which are passed to the procedure of ontology updating. According to the specification, all statistics of the features of semantic relations are stored in the appropriate domain ontology, which provides the ability to self-learning and appropriate adaptability to user needs. The scheme of storage of features is given in Figure 6. Adaptability is provided by weighing of each stored features according to its occurrence during ontology learning. For this purposes every element of a pattern descriptor has its own counter: W_{ml} – for MSR name; S_{ml} – for the concept, which participates in a descriptor as a subject; O_{ml} – for the concept, which participates as an object (see Figure 6). All of them are stored in ontology as an integer type datatype property (see Table 4). This way we can estimate all needed conditional probabilities using only data from the learned ontology.

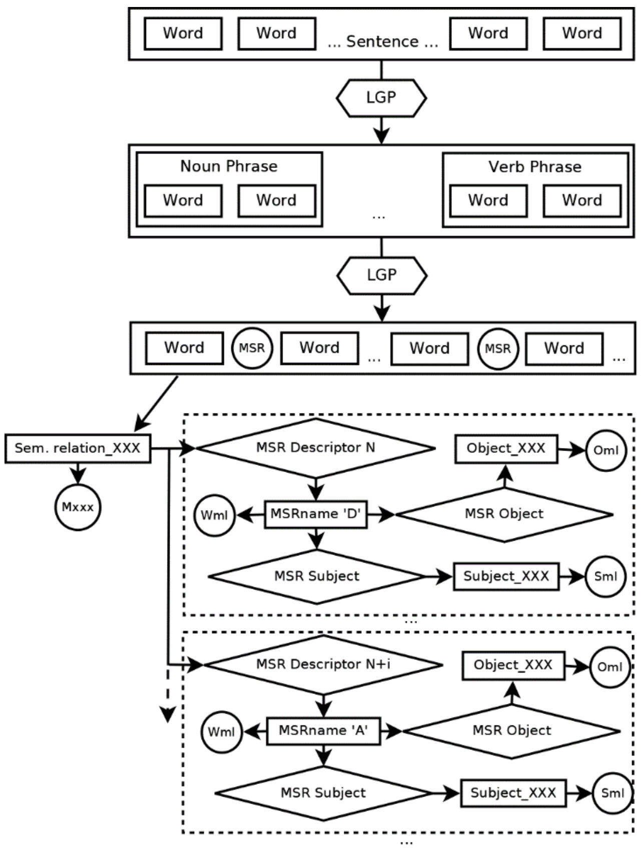


Figure 6. Schema of recorded in ontology the pattern of a semantic relation as a set of descriptors.

A recursive procedure of adding of new concepts to the ontology during its learning was elaborated and applied with help of WordNet API. Each time new word, not represented in the learned ontology by correspondent concept, appears in a learning text the procedure retrieves all (but no more than N most frequent) correspondent synsets from the WordNet database, then retrieves there his hypernym and checks its presence in the ontology. If not, procedure recursively repeats until all kind of the hypernyms of the original word will find their hypernym concept in the learned ontology. For set of obtained concepts the total importance weight W_T of each of them are compared to select the highest one, for which all previously unknown hyponyms up to original word are sequentially added to the ontology as correspondent subclasses.

It is possible thanks to using the method of weighting of ontology concepts and relations, developed by authors [34]. According to this method:

$$W_j^i = Wo_j^i + Ws_j^i + Wn_j^i \qquad (5)$$

1. The total weight W_{Tj}^i of the ontology class is equal to the sum of its own weight Wo_j^i , the weight of the subclasses Ws_j^i and the weight of the adjacent non taxonomic classes Wn_j^i (associated by a semantic relation, other than "is-a"):

where: $Ws_j^i = \sum_k Wc_k^{i+1} L_{j,k}$ – the weight of *k* subclasses of the *j*-th class of the *i*-th level of the taxonomy;

$Wc_k^{i+1} = Wo_k^{i+1} + Ws_k^{i+1}$ – the weight of the class Wc_k^{i+1}; $L_{j,k}$ – the weight of the relation between classes C_j and C_k.

2. At the moment of adding a new subclass to the *i*+1st level, it is assigned its own weight Wo_j^{i+1} equal to half of the class's own weight of the upper (*i*-th) level. The weight of the class Wc_j^i and all parent classes up to the root increases by the weight of the newly created subclass:

$$Wc_j^m = Wc_j^m + Wo_j^{i+1}, \forall m \le i\,. \qquad (6)$$

3. When establishing a relation L_{k_1,k_2} between the concepts *k₁* and *k₂*, an edge appears between the corresponding vertices of the ontology graph, and the weight Wc_2 is added to the weight of the adjacent classes Wn_1, and vice versa, the weight of the new adjacent class Wc_1 is added to Wn_2:

$$Wn_j = \sum_k Wc_k L_{j,k} \qquad (7)$$

4. The weight of an instance in the knowledge base is equal to the total weight (5) of its class in the ontology.

5. The weight of a relation importance is equal to value of its mentions counter (see Figure 7).

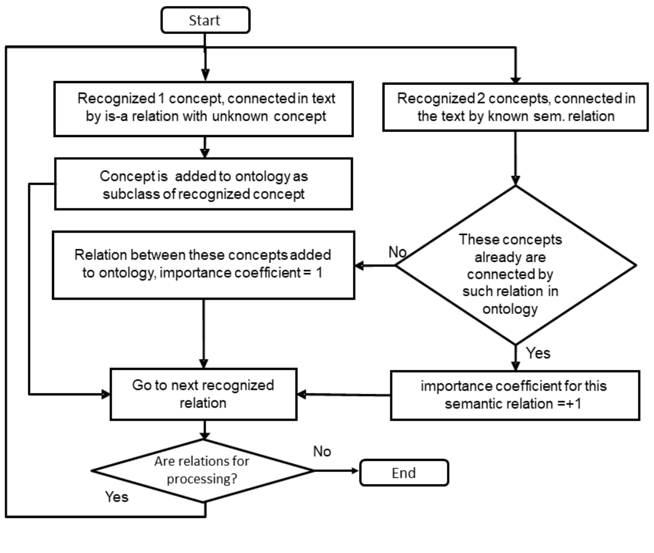

Start
Recognized 1 concept, connected in text by is-a relation with unknown concept
Recognized 2 concepts, connected in the text by known sem. relation
Concept is added to ontology as subclass of recognized concept
Relation between these concepts added to ontology, importance coefficient = 1
No
These concepts already are connected by such relation in ontology
Yes
Go to next recognized relation
importance coefficient for this semantic relation =+1
Are relations for processing?
No
End
Yes

Figure 7. A block-diagram of the algorithm of semantic relations weights updating during ontology learning.

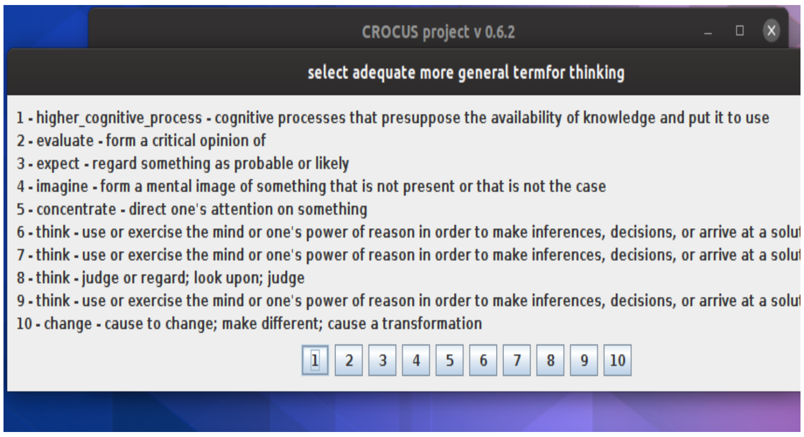
The developed method and correspondent algorithm is the basis for automatic recalculation of the weight of ontology classes, instances and relations between them during its filling and adaptation to a given subject area during operation with the aim to optimize structure and volume of automatically generated ontology [35].

5. IMPLEMENTATION

Ontology learning tools and methods, described above, had been implemented as the software CROCUS (Cognition of Relations Over Concepts Using Semantics).

It is based on Protégé-OWL API and OWL API with functions, realized by LGP API, WordNet API and some other Java libraries. It allows combining work with both OWL1 and OWL2 ontology formats, depending of the task.

The first recognition level – which concept corresponds to a NLT word – is built with help of WordNet: all possible meaning of the word is represented by a set of synsets, which title name and attributes are compared with the names and corresponding attributes of concepts in the ontology [36]. If it is absent CROCUS proposes two options: 1) to choose from the hypernyms, proposed by the WordNet (see: Figure 8), or 2) to let the software choose one automatically based on criteria of most general concept from the subset of known of the set of proposed by the WordNet.



1 - higher_cognitive_process - cognitive processes that presuppose the availability of knowledge and put it to use
2 - evaluate - form a critical opinion of
3 - expect - regard something as probable or likely
4 - imagine - form a mental image of something that is not present or that is not the case
5 - concentrate - direct one's attention on something
6 - think - use or exercise the mind or one's power of reason in order to make inferences, decisions, or arrive at a solut
7 - think - use or exercise the mind or one's power of reason in order to make inferences, decisions, or arrive at a solut
8 - think - judge or regard; look upon; judge
9 - think - use or exercise the mind or one's power of reason in order to make inferences, decisions, or arrive at a solut
10 - change - cause to change; make different; cause a transformation

Figure 8. The window of manual hypernym choosing.

It is possible thanks to automated ontology concept weighting during learning process. A concept is as high at the ontology taxonomy as more general it is. A level of concept place in the taxonomy of the ontology is unambiguously determined by its *Wo* weight. Therefore *Wo* could be interpret as a comparative level of concept generality.

Often a hypernym concept of the looked NLT word is also absent at the ontology. In that case the looking procedure is repeated recursively for each founded hypernym of each synset WordNet representation of the word. As a result of such a search using the WordNet taxonomy, a search tree with a high vertex index is formed (see, for example, the number of hypernyms, suggested by WordNet for the word “thinking” in Figure 8), which leads to the need to calculate the semantic similarity coefficients for the number of concepts equal to the number of terminal vertices of the whole obtained search tree. For reasons of conciseness of the ontology the system chooses the branch of the search tree with the highest *Wo* terminal node parameter. All the non-terminal concept nodes of the search tree branch also then added to the ontology as subclasses in a backward sequence up to the word from the NLT.

All the learned parameters of the ontology, which provide it adaptability to a certain domain and/or user needs, are saved inside this ontology as RDF datatype properties. The place of storage and assignment of parameters is summarized in the Table 4.

Table 4.
Weighting parameters of OWL ontology classes and individuals

Title	Carrier	Carrier address*	Aim
Wo	Class	<Class_name>	Own weight
Ws	Class	<Class_name>	Subclasses weight

Lisa	Class	<Class_name>	Is-a relation weight
Wml	Individual	"default"+<MSR>	MSR counter
Oml	Individual	"default"+<sem.relation>+<MSR>+"object" +<concept>+<POS>	A counter of using a concept as an object in a MSR for a certain semantic relation
Sml	Individual	"default"+<sem.relation>+<MSR>+"subject" +<concept>+<POS>	A counter of using a concept as an subject in a MSR for a certain semantic relation
Mx	Individual	"default"+<sem.relation>	Semantic relation counter

* – all address elements are separated by underline sign

The learning procedure is implemented in LearnSemRelation.java class, which contains the methods:

- findSuperClass();
- addSubClass();
- setDataProperty();
- addPattern();
- readPattern(),

and a set of 15 other auxiliary methods like classExist(), getInstanceMaxNumber(), relationStatisticsView() etc.

The key method addPattern() takes a set of descriptors, produced by the SupplementedLGParser.java class, and performs a sequence of steps:

1) needed standard concepts is adding to the ontology;
2) "default_"-kind auxiliary individuals is adding;
3) learning sentences representative individuals is generated;
4) new semantic relation ontology class is created;
5) correspondent default-individual is created with the datatype RDF property *Mx*=1;
6) an abstract MSR concept is created or updated;

For each obtained descriptor (see Figure 6):

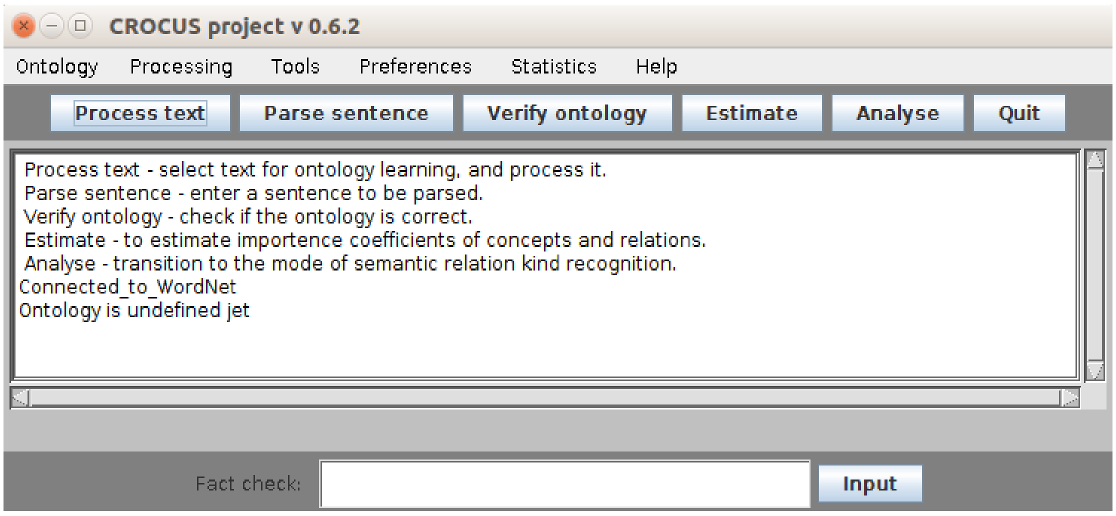
- consequently MSR subject and object concepts are adding to an ontology recursively (as described above) and/or, if any of them already exists, default auxiliary individuals, mentioned in Table 4, are created;
- Wml, Oml and Sml datatype properties of the correspondent individuals are updating;
- importance weights Wo, Ws, Wn are recalculating with accordance with (5)…(7) to choose right superclass during next learning steps.

A set of all needed functions are implemented in the CROCUS system. The main user interface of the system is presented on the Figure 9. The most commonly used functions of the system are moved to the toolbox of the front panel:

- Process text is a complex procedure of ontology learning by parsing text using Link Grammar parser as a set of sentences with the same semantic relation which should be sequentially processed recursively updating the ontology with help of WordNet lexical database;
- Parse sentence is a similar to text processing procedure except it is manual and under one separate sentence;
- Verify ontology opens separate dialog interface with possibility of observation of all main values and parameters of the learned adaptive ontology: concepts, relations, weights, properties values etc.;
- Estimate is a procedure of forced up-down recalculation of concept weights;

- Analyze – is a mode of automated recognition of semantic relation using Bayesian classifier, described above.

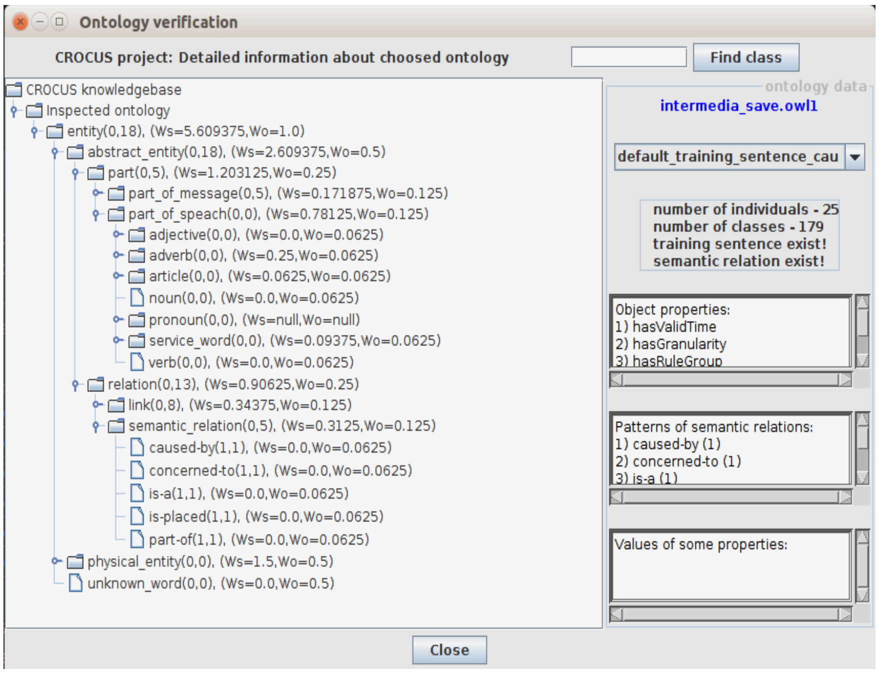
According to the main aim of the software CROCUS it assists in automated and semi-automated ontology learning by natural language text on the level of semantic relations, terminological TBox axioms and facts/predicates. A learned adaptive ontology processed by such learning procedures will be ready to distinguish semantic meaning of the NLT sentences without an expert participation.



```
Process text - select text for ontology learning, and process it.
Parse sentence - enter a sentence to be parsed.
Verify ontology - check if the ontology is correct.
Estimate - to estimate importence coefficients of concepts and relations.
Analyse - transition to the mode of semantic relation kind recognition.
Connected_to_WordNet
Ontology is undefined jet
```

Figure 9. CROCUS main interface.

Current state of a learned ontology can be monitored in a separate window, shown at the Figure 10. There are hierarchical structure of the ontology, weights of the learned concepts *Wo* and *Ws*, evaluated by the method, mentioned above on section 4, main ontology statistics etc.



```
CROCUS knowledgebase
  Inspected ontology
    entity(0,18), (Ws=5.609375,Wo=1.0)
      abstract_entity(0,18), (Ws=2.609375,Wo=0.5)
        part(0,5), (Ws=1.203125,Wo=0.25)
          part_of_message(0,5), (Ws=0.171875,Wo=0.125)
          part_of_speach(0,0), (Ws=0.78125,Wo=0.125)
            adjective(0,0), (Ws=0.0,Wo=0.0625)
            adverb(0,0), (Ws=0.25,Wo=0.0625)
            article(0,0), (Ws=0.0625,Wo=0.0625)
            noun(0,0), (Ws=0.0,Wo=0.0625)
            pronoun(0,0), (Ws=null,Wo=null)
            service_word(0,0), (Ws=0.09375,Wo=0.0625)
            verb(0,0), (Ws=0.0,Wo=0.0625)
        relation(0,13), (Ws=0.90625,Wo=0.25)
          link(0,8), (Ws=0.34375,Wo=0.125)
          semantic_relation(0,5), (Ws=0.3125,Wo=0.125)
            caused-by(1,1), (Ws=0.0,Wo=0.0625)
            concerned-to(1,1), (Ws=0.0,Wo=0.0625)
            is-a(1,1), (Ws=0.0,Wo=0.0625)
            is-placed(1,1), (Ws=0.0,Wo=0.0625)
            part-of(1,1), (Ws=0.0,Wo=0.0625)
      physical_entity(0,0), (Ws=1.5,Wo=0.5)
      unknown_word(0,0), (Ws=0.0,Wo=0.5)

intermedia_save.owl1
default_training_sentence_cau

number of individuals - 25
number of classes - 179
training sentence exist!
semantic relation exist!

Object properties:
1) hasValidTime
2) hasGranularity
3) hasRuleGroup

Patterns of semantic relations:
1) caused-by (1)
2) concerned-to (1)
3) is-a (1)

Values of some properties:
```

Figure 10. Ontology parameters monitoring interface of the CROCUS system. An initial state of the ontology.

6. EXPERIMEMTAL RESULTS

An almost empty ontology had been used to test the process of ontology learning from text using the Link Grammar parser and WordNet API. An initial metrics, provided by the Protégé 3.4. is presented at the Figure 11.

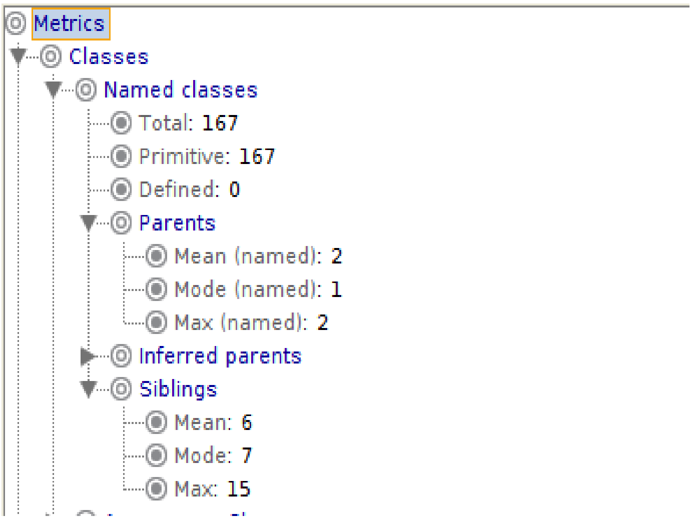

Metrics
Classes
Named classes
Total: 167
Primitive: 167
Defined: 0
Parents
Mean (named): 2
Mode (named): 1
Max (named): 2
Inferred parents
Siblings
Mean: 6
Mode: 7
Max: 15

Figure 11. A basic metrics of the initial ontology.

Learning texts had been selected from a corpus of scientific paper abstracts in the common Material Science and Computer Engineering domain. Random first 20 sentences which using each learned semantic relation had been adopted and applied to learn an ontology to recognize the 3 kind of relations:

• consist-of;
• is-a;
• shows.

The selection of semantic relation kind was random with taking into account the statistics of their using in the domain abstracts: in the middle of 1/3 top popular relations (see: Figure 12).

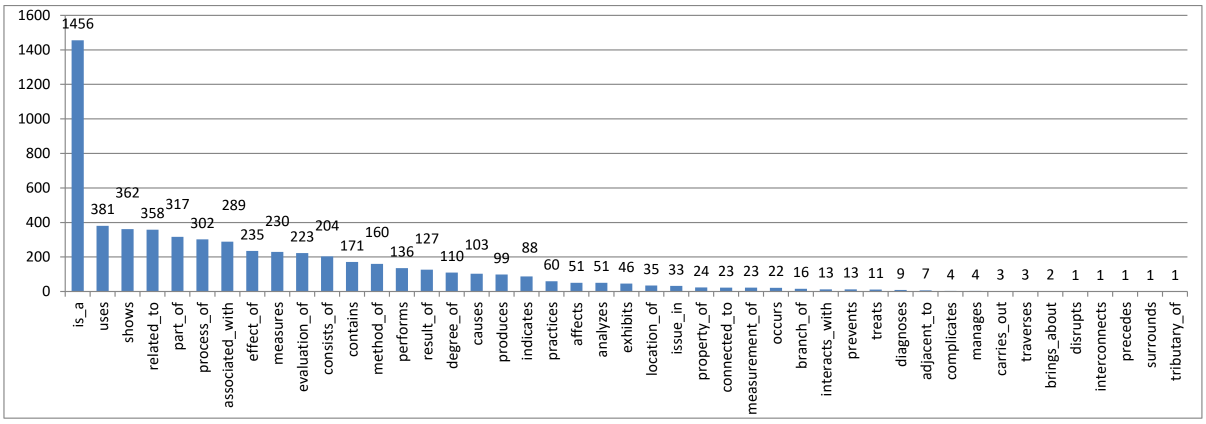
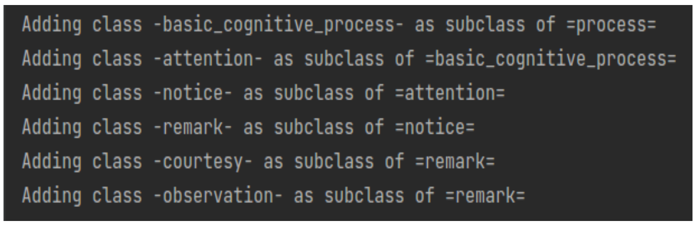


Figure 12. Histogram of semantic relations occurrence in the used text corpus.

When a new word, not represented in the ontology by a correspondent concept, appeared in a learning text the recursive procedures of linking it to an ontology hypernym (findSuperClass-addSubClass), mentioned in the previous section, were applied.

Despite choosing the highest Wo parameter of the hypernym candidate to be a superclass i.e. most general concept from synsets, proposed by the WordNet taxonomy, the algorithm builds long hierarchy chains for each new word (see, for example, Figure 13).



```
Adding class -basic_cognitive_process- as subclass of =process=
Adding class -attention- as subclass of =basic_cognitive_process=
Adding class -notice- as subclass of =attention=
Adding class -remark- as subclass of =notice=
Adding class -courtesy- as subclass of =remark=
Adding class -observation- as subclass of =remark=
```

Figure 13. The backward chain of the word 'observation' hypernyms, added as subclasses to existing in the ontology superclass 'process'.

During a learning process the number of superclasses, which should be added to the ontology simultaneously (previously) with the new concept from the learned text gradually decreases, like it could be seen on the Figure 14. On the Figure 15 it is presented the data of learned descriptors on the example of "is-a" semantic relation pattern, structure of which was shown at the Figure 6. The correspondent numerical data for the learned three kind of relations after learning by 20 sentences for each is provided in the Table 5.

Table 5. Number of learned pattern descriptors

Sem. relation	Desc 1	Desc 2	Desc 3	Desc 4	Desc 5	Desc 6	Desc 7	Desc 8	Desc 9	Desc 10
Consist-of	27	19	13	8	6	3	3	3	2	2
Is-a	22	11	9	8	6	5	4	4	4	3
Shows	22	12	12	12	4	4	3	3	3	3

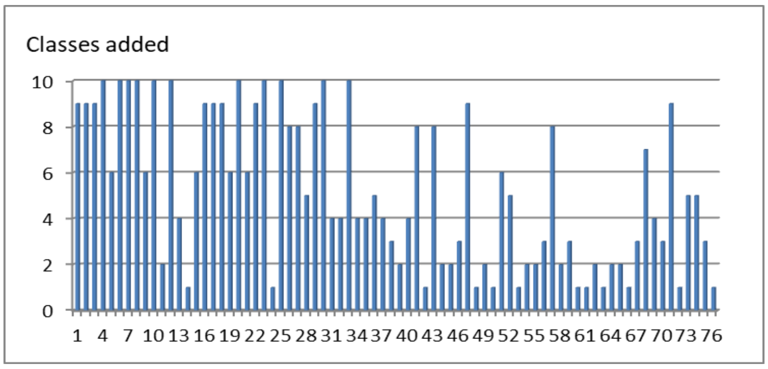


Figure 14. Number of superclasses of the learned word, added from the WordNet to the ontology on each sequential step of the ontology learning procedure.

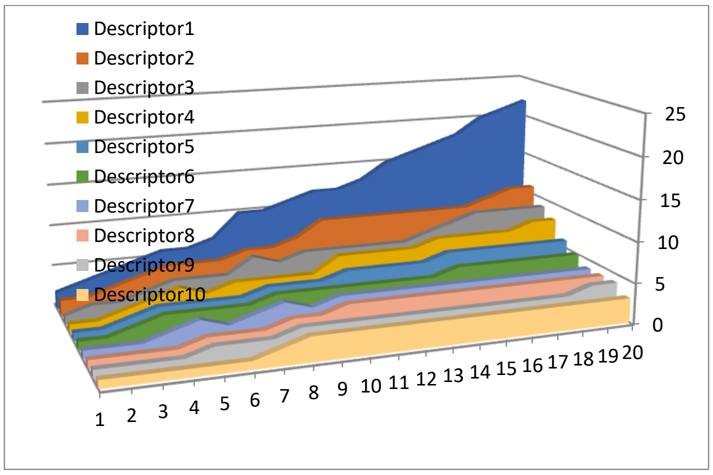
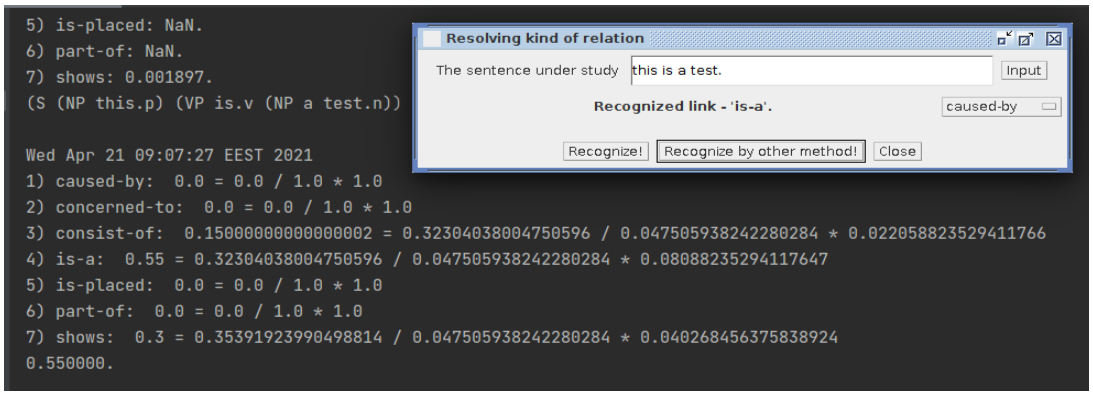


Figure 15. Increasing number of "is-a" pattern descriptors during the process of learning by 20 sentences.

Testing classification experiments had been executed with aim to examine the developed software and the algorithm, described above in previous section, based on the naïve Bayes classifier and the set of learned semantic relation pattern descriptors, mentioned in the Table 5. The results are presented as a print-screen copy of the results of the working CROCUS software (see Figure 16).



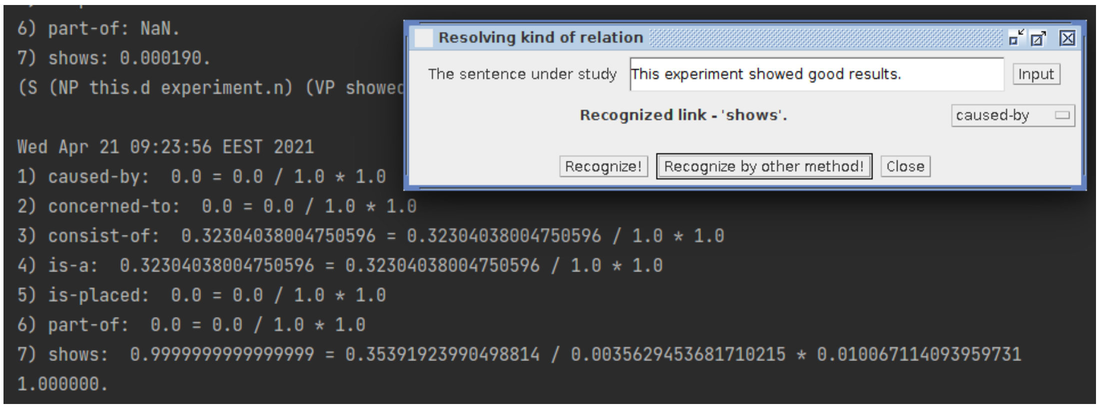
Resolving kind of relation

The sentence under study: this is a test.

Recognized link - 'is-a'.

```
5) is-placed: NaN.
6) part-of: NaN.
7) shows: 0.001897.
(S (NP this.p) (VP is.v (NP a test.n))

Wed Apr 21 09:07:27 EEST 2021
1) caused-by:  0.0 = 0.0 / 1.0 * 1.0
2) concerned-to:  0.0 = 0.0 / 1.0 * 1.0
3) consist-of:  0.15000000000000002 = 0.32304038004750596 / 0.047505938242280284 * 0.022058823529411766
4) is-a:  0.55 = 0.32304038004750596 / 0.047505938242280284 * 0.08088235294117647
5) is-placed:  0.0 = 0.0 / 1.0 * 1.0
6) part-of:  0.0 = 0.0 / 1.0 * 1.0
7) shows:  0.3 = 0.35391923990498814 / 0.047505938242280284 * 0.040268456375838924
0.550000.
```



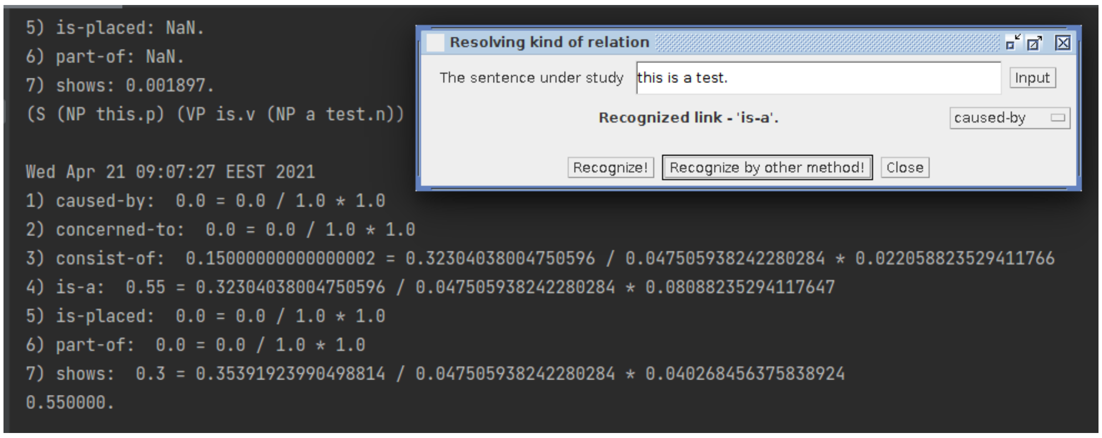
Resolving kind of relation

The sentence under study: This experiment showed good results.

Recognized link - 'shows'.

```
6) part-of: NaN.
7) shows: 0.000190.
(S (NP this.d experiment.n) (VP showed

Wed Apr 21 09:23:56 EEST 2021
1) caused-by:  0.0 = 0.0 / 1.0 * 1.0
2) concerned-to:  0.0 = 0.0 / 1.0 * 1.0
3) consist-of:  0.32304038004750596 = 0.32304038004750596 / 1.0 * 1.0
4) is-a:  0.32304038004750596 = 0.32304038004750596 / 1.0 * 1.0
5) is-placed:  0.0 = 0.0 / 1.0 * 1.0
6) part-of:  0.0 = 0.0 / 1.0 * 1.0
7) shows:  0.9999999999999999 = 0.35391923990498814 / 0.0035629453681710215 * 0.010067114093959731
1.000000.
```



Resolving kind of relation

The sentence under study: this is a test.

Recognized link - 'is-a'.

```
5) is-placed: NaN.
6) part-of: NaN.
7) shows: 0.001897.
(S (NP this.p) (VP is.v (NP a test.n))

Wed Apr 21 09:07:27 EEST 2021
1) caused-by:  0.0 = 0.0 / 1.0 * 1.0
2) concerned-to:  0.0 = 0.0 / 1.0 * 1.0
3) consist-of:  0.15000000000000002 = 0.32304038004750596 / 0.047505938242280284 * 0.022058823529411766
4) is-a:  0.55 = 0.32304038004750596 / 0.047505938242280284 * 0.08088235294117647
5) is-placed:  0.0 = 0.0 / 1.0 * 1.0
6) part-of:  0.0 = 0.0 / 1.0 * 1.0
7) shows:  0.3 = 0.35391923990498814 / 0.047505938242280284 * 0.040268456375838924
0.550000.
```

Figure 16. Using naïve Bayes classifier and the same set of learned pattern descriptors for recognition a kind of semantic relation in testing NLT sentences.

7. CONCLUSION

In this study, the system of automated and semi-automated ontology learning form text had been developed using Carnegie Mellon Link Grammar Parser and WordNet API. A combination of two different methods to distinguish semantic relations in NLT sentences had been applied: analysis of the sentence constituent trees – by means of Link Grammar Parser for explicit meta-semantic relations recognition and Naïve Bayes supervised learning – for direct recognition semantic relations with help of additional meta-semantic data, obtained on the previous step. Developed approach was implemented in the Java desktop application using WordNet API, OWL API and Protégé-OWL API. Experimental results were compared to expert analysis and had shown good recognition reliability.

REFERENCES

[1] Manning, C. D., Raghavan, P., & Schultze, H. (2008). Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK.

[2] Ivokhin, E.V., Oletsky, O.V. Restructuring of the Model "State–Probability of Choice" Based on Products of Stochastic Rectangular Matrices. Cybern Syst Anal. Vol.58 (2), pp.242-250 (2022). https://doi.org/10.1007/s10559-022-00456-z

[3] Toward an Architecture for Never-Ending Language Learning. A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr. and T.M. Mitchell. In Proceedings of the Conference on Artificial Intelligence (AAAI), 2010.

[4] Mitchell, T. & Kisiel, B. & Krishnamurthy, J. & Lao, Ni & Rivard, Kathryn & Mohamed, T. & Nakashole, N. & Platanios, Emmanouil Antonios & Ritter, A. & Samadi, M. & Settles, B. & Cohen, W. & Wang, Richard & Wijaya, D. & Gupta, A. & Chen, Xinlei & Saparov, A. & Greaves, M. & Welling, J. & Gardner, M.. (2018). Never-ending learning. Communications of the ACM. 61. 103-115. 10.1145/3191513.

[5] Russell, S. Human Compatible: Artificial Intelligence and the Problem of Control.- Penguin Publishing Group. – 352 p. – 2019

[6] Yates, A., Banko, M., Broadhead, M., Cafarella, M., Etzioni, O., Soderland, S. TextRunner: Open Information Extraction on the Web. Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), Pp. 25-26, 2007.

[7] Behnke, G., Bercher, P., Biundo, S., Glimm, B., Ponomariov, D., Schiller, M.: Integrating ontologies and planning for cognitive systems. In: Proceedings of the 28th International Workshop on Description Logic (DL 2015). CEUR Workshop Proceedings (2015)

[8] France-Mensah, Jojo, and William J. O'Brien. "A Shared Ontology for Integrated Highway Planning." Advanced Engineering Informatics 41 (2019): n. pag. Advanced Engineering Informatics. Web.

[9] Lucas Martínez, N.; Martínez-Ortega, J.-F.; Castillejo, P.; Beltrán Martínez, V. Survey of Mission Planning and Management Architectures for Underwater Cooperative Robotics Operations. Appl. Sci. 2020, 10, 1086.

[10] Shannon, C. E. (2001). A mathematical theory of communication. ACM SIGMOBILE mobile computing and communications review, 5(1), 3-55.

[11] Brunstein, Ada "Annotation Guidelines for Answer Types". LDC Catalog. Linguistic Data Consortium, 2002,

[12] Sekine, S., Kiyoshi Sudo and Chikashi Nobata. "Extended Named Entity Hierarchy." LREC (2002).

[13] Ritter, A.; Clark, S.; Mausam; Etzioni., O. (2011). Named Entity Recognition in Tweets: An Experimental Study (PDF). Proc. Empirical Methods in Natural Language Processing

[14] Derczynski, Leon and Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphael Troncy, Johann Petrak, and Kalian Botcheva (2014). «Analysis of named entity recognition and linking for tweets». Information Processing and Management 51(2): Pp. 32-49.

[15] Taylor A., Marcus M., Santorini B. (2003) The Penn Treebank: An Overview. In: Abeillé A. (eds) Treebanks. Text, Speech and Language Technology, vol 20. Springer, Dordrecht. https://doi.org/10.1007/978-94-010-0201-1_1

[16] Sennrich, R., Haddow, B., and Birch, A., "Neural Machine Translation of Rare Words with Subword Units", <i>arXiv e-prints</i>, 2015.

[17] Miller G.A. Wordnet: A lexical database for English / G.A. Miller // Communications of the ACM Vol. 38, No. 11. – 1995. – Pp. 39–41.

[18] Salton, G., Wong, A. and Yang, C. S.. 1975. A vector space model for automatic indexing. Commun. ACM 18, 11 (Nov. 1975), 613–620.

[19] Anh-Dung Vo, Quang-Phuoc Nguyen, and Cheol-Young Ock. 2018. Automatic Knowledge Extraction for Aspect-based Sentiment Analysis of Customer Reviews. In Proceedings of the

10th International Conference on Computer Modeling and Simulation (ICCMS 2018). Association for Computing Machinery, New York, NY, USA, 110–113.

[20] Cambria, E; Schuller, B; Xia, Y; Havasi, C (2013). "New avenues in opinion mining and sentiment analysis". IEEE Intelligent Systems. 28 (2): 15–21. CiteSeerX 10.1.1.688.1384

[21] Bloomfield, Leonard. 1933. Language. New York: Henry Holt ISBN 0-226-06067-5, ISBN 90-272-1892-7

[22] Chomsky, Noam 1957. Syntactic Structures. The Hague/Paris: Mouton.

[23] Jurafsky D and Martin JH. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition 2009. Upper Saddle River, NJ: Pearson Prentice Hall, 2009

[24] Schmitt, X., Kubler, S., Robert, J., Papadakis, M., LeTraon, Y. A Replicable Comparison Study of NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate. Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), 2019, Pp.338-343.

[25] Carlson, L., Marcu, D. and Okurowski, M.(2003), Building a discourse-tagged corpus in the framework of rhetorical structure theory, in J. van Kuppevelt and R. Smith (eds), Current Directions in Discourse and Dialogue, Kluwer Academic Publishers, pp. 85–112.

[26] Lin, D.(1998), Automatic retrieval and clustering of similar words, Proceedings of the 17th international conference on Computational linguistics, Association for Computational Linguistics, Morristown, NJ, USA, pp. 768–774.

[27] Juan Soler-Company, Leo Wanner, On the role of syntactic dependencies and discourse relations for author and gender identification, Pattern Recognition Letters, Vol. 105, 2018, Pp. 87-95.

[28] Bohnet, B., Nivre, J., A transition-based system for joint part-of-speechtagging and labeled non-projective dependency parsing, in: Proceedingsof the 2012 Joint Conference on Empirical Methods in Natural LanguageProcessing and Computational Natural Language Learning, Association for Computational Linguistics. 2012. Pp. 1455–1465.

[29] Cunningham, H. GATE, a General Architecture for Text Engineering. Computers and the Humanities 36, 223–254 (2002).

[30] Bontcheva, K., Tablan, V., Maynard, D., Cunningham, H. Evolving GATE to Meet New Challenges in Language Engineering. Natural Language Engineering. 10 (3/4), Pp. 349-373. 2004.

[31] Fundel-Clemens, Katrin & Küffner, Robert & Zimmer, Ralf. (2007). RelEx – Relation extraction using dependency parse trees. Bioinformatics (Oxford, England). 23. 365-71. 10.1093/bioinformatics/btl616.

[32] Sleator, Daniel & Temperley, David. (1995). Parsing English with a Link Grammar. CoRR. abs/cmp-lg/9508004.

[33] Maynard, D., Bontcheva, K., & Augenstein, I. (2016). Natural language processing for the semantic web. Synthesis Lectures on the Semantic Web: Theory and Technology, 6(2), 1–194.

[34] Lytvyn V., Vysotska V., Dosyn D., Burov Y. Method for ontology content and structure optimization, provided by a weighted conceptual graph // Webology. 2018. Vol. 15, Iss. 2. P. 66–85.

[35] Problems of Ontology Structure and Meaning Optimization and Theirs Solution Methods / Vasyl Lytvyn, Oksana Oborska, Victoria Vysotska, Dmytro Dosyn, Andriy Demchuk, Yevhen Burov, Petro Kravets, Nazar Oleksiv // Computational linguistics and intelligent systems : proceedings of the 4nd International conference, 23-24 April 2020, Lviv, Ukraine. — Lviv : Lviv Politechnic Publishing House, 2020. — Vol 2 : Proceedings of the 4nd International conference, COLINS 2020. Workshop, Lviv, Ukraine April 23-24, 2020. — P. 21–40.

[36] O.Oletsky. On Constructing Adjustable Procedures for Enhancing Consistency of Pairwise Comparisons on the Base of Linear Equations. CEUR Workshop Proceedings, 2021, 3106, pp. 177–185