Unsupervised Language Learning in OpenCog

Alex Glushchenko[1], Andres Suarez[2], Anton Kolonin[1](✉), Ben Goertzel[1,2], Claudia Castillo[2], Man Hin Leung[2], and Oleg Baskov[1]

[1] SingularityNET Foundation, Amsterdam, Netherlands
{anton,ben}@singularitynet.io
[2] Hanson Robotics, Hong Kong, China

Abstract. We discuss technology capable to learn language without supervision. While the entire goal may be too ambitious and not achievable to full extent, we explore how far we can advance grammar learning. We present the current approach employed in the open source OpenCog Artificial Intelligence Platform, describe the cognitive pipeline being constructed and present some intermediate results.

Keywords: Categorization · Clustering · Computational linguistics
Dimensionality reduction · Formal grammar · Grammar induction
Natural language processing · Unsupervised learning · Vector space

1 Introduction

This work is driven by a desire to make sense of the possible mechanisms underlying natural intelligence and applied to language-based communication skills. According to earlier work [1–3], human intelligence is substantially connected with language acquisition abilities. As pointed out in [1], most of such acquisition is loosely supervised, while existing machine learning techniques require more effort and training data to reach the level of human children.

Another point being made in [2] is that current natural language processing applications require a formal grammar or an annotated corpora as input. In practice, however, the cost of creating such grammars results in a lack of good-quality ones for many languages. Indeed, several years of human effort are required to create a formal grammar for a language, given formalisms such as Link Grammar (LG) [4]. While this has been done to great extent in English, as the language most widely used in electronic communications, other languages are far less represented, and for many natural languages such effort has never been undertaken. Even for English, the existent grammar dictionary in Link Grammar can only handle literary English texts and fails to support deviations, like dialects used in chat rooms, or domain-specific jargon. Our work, aiming to automatically produce a grammar from unannotated text, could potentially reduce the effort needed to process any language and make it comprehensible by software.

Moreover, many natural language processing (NLP) applications for text mining and information extraction rely on pattern-based approaches for classification, entity extraction and attribution [5]. For such applications, it is crucial to have a way to identify textual patterns which could be used for entity extraction, as well as for finding

M. Iklé et al. (Eds.): AGI 2018, LNAI 10999, pp. 109–118, 2018.
https://doi.org/10.1007/978-3-319-97676-1_11

relationships between entities. That is, the patterns should be flexible enough to represent different textual representations of the same semantic entity as well as to describe patterns of connections between such entities represented in text. Technology that infers these patterns in an unsupervised way, given prepared (controlled) corpora, could be highly valuable, as it would make NLP applications cheaper, faster, more precise and efficient.

A system like the one we propose would also help approach the artificial general intelligence problem known as “Baby Turing Test” [6]. While the classic “Turing Test” expects an AI system to display conversational intelligence comparable to a human, it does not prevent such system from being simply hardcoded, and does not guarantee that the system actually learned its abilities through education and interaction with its environment. In turn, the “Baby Turing Test” requires that an “uneducated” system undergoes the process of experiential or didactic training, and eventually develop skills sufficient to pass the classic “Turing Test”. Our work could help on this goal, at least from a comprehension perspective, so that an artificially intelligent system could be incrementally fed with information in a given language, and eventually make sense of this information and new information of similar kind, like texts in the known language within the same knowledge domain.

The overall direction of research is shaped by earlier works [2, 3], and it’s based on representing linguistic structures as “sheafs” of graphs [7], where the elementary structures of the graphs are represented with so-called “disjuncts” from Link Grammar [4], which can then be used to infer a grammar in Link Grammar format. The input of the grammar-learning process are statistical parses generated with a minimum spanning tree (MST) approach based on “mutual information” computed for co-occurring words in sentences [8].

In this work, we discuss some practical aspects of implementing the NLP pipeline for unsupervised language learning (ULL), including building a vector space [9], its dimension reduction (DR), and unsupervised category learning grammatical and semantic concepts by means of clustering [10–15]. We will also consider the different approaches for word-sense disambiguation (WSD) applied [16, 17].

All of the research and developments discussed further are performed in the scope of the open source OpenCog project [18] and SingularityNET platform.

2 Background

For the work discussed in this article we are making certain assumptions and considering specific options, as discussed below. We are not sure if unsupervised learning on unannotated corpora with neither grounding nor reinforcement feedback can succeed at all. Still, we do want to advance in this direction to see if we can learn at least most of the grammar and some of the semantics. We also understand that there is no clear boundary between grammatical and semantic categories, because certain semantics categories such as time, gender and plurality may affect grammar to extents specific to particular languages. To make the problem solvable at least to some extent, we can make certain simplifications and relaxations, as follows.

Controlled Corpora. Although unannotated, our learning corpora are pre-processed to reduce the amount of gibberish (tables, graphics, mixed languages, etc.) found in them, as they are obtained from public sources. We also use corpora of different complexity in the following order, targeting different goals.

- **Proof-of-Concept Turtle** (POC-Turtle) corpus - represents closed semantic space of very few words communicated in simplex Turtle language [19], used in semantic web programming, with complexity limited to three words per sentence in strict Subject-Verb-Predicate triplet grammar. Complexity of such language can be thought closer to complexity of language that non-human primates can learn [20] or that children at age up to 3 can use [21].
- **Proof-of-Concept English with no ambiguity** (POC-English-NoAmb) corpus - manually created closed semantic space of very few English words with nearly the same frequency of use, communicated in simple grammatical construction of 4–5 words per sentence, without any ambiguous words.
- **Proof-of-Concept English with ambiguity** (POC-English-Amb) corpus – similar to the above, but with two words involved in semantically and grammatically ambiguous constructions. Semantic ambiguity is represented by word “board” which can be either board of a ship or black board. Grammatical ambiguity is represented by word “saw” which can be either noun or past form of verb “see”.
- **Child Directed Speech** corpus - collection of English communications directed to children with adapted lexicon and grammar complexity.
- **Gutenberg Children and Adult** corpora - children and adult subsets of literary corpora collected within Gutenberg Project (see https://www.gutenberg.org/).

Incremental Learning and “One-Shot Learning”. Incremental learning approach may be interesting from a number of perspectives. There are different points of view whether using simplified language when conversing to child can advance language learning or complicate it [1]. In real life the richness of lexicon increases gradually over years of child development. So we would like to use both approaches. In one approach called “One-shot Learning”, we would try to have entire grammar learned at one upon one successful reading of entire corpus. In the other one called “Incremental Learning”, we would split the corpus into sections with gradually increasing complexity (in terms of either maximum sentence length of richness of lexicon or both), trying to capture more grammatical constructions incrementally. During the second approach we would measure the learning curve tracking the ability of developing system to pass the Baby Turing Test [6].

Symbolic vs. Sub-symbolic Approach. We are not limiting ourselves to use either “old school” symbolic approach such as LG [4] or “new school” of distributional representations in NLP with its latest advances [9]. In fact, we are planning to try both and it is anticipated that the final solution will be a combination of the two.

Variety of Vector Spaces. Generally, the word space can be represented with vector embeddings created by a number of sub-symbolic approaches. The most widely used vector space is the space of words, either “Bag-of-Words” or “Skip-gram” [9]. In our work we study the ways of replacing word tokens with word senses defined earlier. We also introduce the space of connectors - directed connectors between words, so the word "like" in phrases "like you" and "you like" would form two different dimensions "like-" and "like+" following the Link Grammar notation [2, 4]. Finally, we will also introduce the space of disjuncts so the word "like" in phrases "I like you" and "you like me" would form two dimensions corresponding to the disjuncts "I- & you+" and "you- & me+". Notably, the space of words is the most dense, the space of connectors is up to two times more sparse, and the space of disjuncts might appear orders of magnitude more sparse on large corpora.

Disambiguation. Since we are going to use statistical parsing, the question arises - how to compute mutual information for ambiguous words. In the sentence "I saw the saw", we can try disambiguating "saw" into "saw@noun" and "saw@verb" prior to calculating the MI, thus facilitating the parser task. The alternative is to parse the text as is, and later try to find the different senses of each word. We will try both options.

Mutual Information Counting. Another question is whether mutual information (MI) should be direction-sensitive, so we count directed links between words [4], or direction-insensitive, so we count co-occurring word pairs no matter what their mutual positions are. We are counting directed links but co-occurrence counting is possible as option to try.

Morphology. The importance of morphology in language comprehension and learning is well understood and there are approaches known to handle that [2, 3]. However, for now we do not consider this level of complexity. We deal with word tokens as entire symbols, disregarding their internal morphology and potential token interactions.

3 Natural Language Pipeline Architecture

The general overview of the cognitive pipeline architecture for the current stage of the project is presented below. Further, we describe specific the components, with their options. In the current implementation the pipeline is linear, so no iterative loops can take place at the moment. The entire pipeline with most of components is being developed as open source at https://github.com/singnet/language-learning with TextParser is being maintained as part of OpenCog at https://github.com/opencog/opencog/.

1. Text Pre-Cleaner - preprocesses corpus files with configurable cleanup and normalization options (is implemented now).
2. Sense Pre-Disambiguator - optionally, performs word disambiguation and builds senses from tokens (is being implemented now).
3. Text Parser - parses sentences of word tokens or senses with one of the possible approaches (is implemented now but may be improved in the future).
4. Grammar Learner - learns word categories from parses, infers grammar in LG format (is implemented now and improvements are ongoing).
5. Tester/Evaluator - evaluates quality of inferred grammar (is implemented).

Text Pre-cleaner. The goal of this component is to standardize corpora tests, removing HTML markup and graphics, normalizing different varieties of the same punctuation marks, interpreting UTF codes, and optionally converting all characters to lowercase. Although there is controversy about representing the capitalized and lower-cased versions of a word as the same token (e.g. "You" and "you"), we currently proceed with converting all texts to lower-case, on the basis that capitalization is not expressed explicitly in spoken conversations, and that all words have the same pronunciation regardless of their location in a sentence. This avoids a single meaning to be represented as two distinct vectors, one for each of its capitalized and non-capitalized versions. We may move past this point in future work and represent them individually.

Sense Pre-disambiguator. In order to use appropriate word-sense entries for mutual information counting during statistical parsing, we may try to disambiguate word tokens before the parsing takes place. For doing so, we may use distributed representation of words with n-grams and skip-grams in vector space of adjacent words [9] using AdaGram to provide word-sense disambiguation [17]. AdaGram extends the renowned Skip-gram methodology of word2vec [9] to include one vector embedding per word-sense, without fixing a number of senses per word a priori.

Text Parser. This component implements two phases: Mutual Information (MI) counting, with the Observer sub-component, and Minimum Spanning Tree (MST) Parser, accordingly to earlier design [3, 8].

Mutual information calculation during the Observer phase may be implemented in one of four ways: (a) cartesian combination of per-sentence words without account for distance; (b) cartesian combination of per-sentence words with account for distance; (c) sampling all parses produced by the Link Grammar Parser in random parsing mode; (d) sampling limited number of parses from the Link Grammar Parser in random parsing mode.

The pointwise mutual information (PMI), also known as focus mutual information (FMI) or association ratio [12] for an ordered pair of words *(x, y)* is a measure of the level of association of the two words in a given context, and is computed as:

$$PMI(x, y) = log2(p(x, y)/(p(x) \cdot p(y)))$$

where *p(x), p(y)*, and *p(x,y)* are short for $P(X = x)$, $P(Y = y)$ and $P(X = x,\ Y = y)$ respectively. Here *X* the is the random variable of the event of watching a word *x* to the left of any other word in a sentence, i.e. the probability of observing the ordered pair *(x, *)*. Similarly, Y is the random variable of the event of watching a word y to the right of any other word in a sentence, i.e. the probability of observing the ordered pair *(*, y)*. Thus *p(x, y)* is the probability of observing the ordered pair *(x, y)* in a sentence. In turn this probabilities are calculated as:

$$\begin{aligned} p(x) &= N(x, *)/N(*, *) \\ p(x) &= N(*, y)/N(*, *) \\ p(x, y) &= (x, y)/N(*, *) \end{aligned}$$

Notice tha $N(x, y)$ is not necessarily the same as $N(y, x)$. PMI is a ratio that compares the probability of observing the two words together in a specific order vs observing them separately, and therefore lies in the range $(-\infty, \infty)$.

The way $N(x, y)$ is counted (the number of appearances of the pair (x, y) in the corpus) depends on choice from the above-mentioned methods. For cartesian combination methods (options a and b above), the pair (x, y) is counted only if x and y occur within distance R (a parameter) in the current sentence (distance r is defined as difference in word position in the sentence: $r = pos(y) - pos(x)$). When disregarding distance, they are counted one time per co-occurrence: $N(x, y) = \sum r < R(1)$, across all appearances of x and y in each of the sentences in the corpus. When accounting for distance, we count the pair R/r times: $N(x, y) = \sum r < R(R/r)$, so the words in greater proximity are getting more counts, with default count as R instead of 1.

For counting methods c and d above, the LG Parser can produce an exhaustive set of possible parses for a sentence, regardless of any grammar or prior knowledge on relationships between words. We can consider all possible parses if sentence length is small (method c), or select a number N (a parameter) of randomly chosen parses, for longer sentences where number of possible trees bursts exponentially. In these methods the pair (x, y) is counted each time x and y are linked together in a parse tree for the given sentence.

Once mutual information is collected, our MST-parser approximates the spanning tree with highest total MI [8], and returns that as output. A tree's MI score is computed as the sum of all linked word pair scores, where score is mutual information per word pair. In this step, we also test if accounting for distance in different ways improves the resulting parses: (i) $Score = PMI * R/r$; (ii) $Score = PMI + 1/r$; (iii) $Score = PMI + R/r$, with r and R as defined above.

Grammar Learner. This pipeline component processes the parse trees produced by the Text Parser in two phases: Category Learning and Grammar Induction. The Category Learning phase includes Vector Space modeling, Clustering, and optional Generalization sub-phases.

The Vector Space dimensions are chosen from: (a) words - either word tokens or word senses; (b) connectors [3, 4]; (c) disjuncts [3, 4]. Positive pointwise mutual information (PPMI) [12] is used for term weighting.

Clustering is performed using "sub-symbolic" or "symbolic" approach. The "sub-symbolic" Unsupervised Category Learning includes dimensionality reduction (DR) with singular value decomposition (SVD) [13] and K-means clustering [14]. The optimal number of clusters is selected based on maximum Silhouette index [15] value.

The alternative "symbolic" approach to clustering the disjunct space implies consecutive merging the single-germ-single-disjuncts ("seeds") extracted from the parse tree into single-germ-multi-disjunct "stalks" and multi-germ-multi-disjunct lexical entries.

The optional generalization agglomerates the learned categories (clusters) into higher-level grammatical categories, preserving relationships between child and parent category clusters.

Grammar Induction infers grammar links between the learned categories (clusters) by statistical processing the parse tree. The Link Grammar rules are induced for the learned set of clusters as either (a) sets of connectors to the linked clusters or (b) sets of disjuncts consisting of connectors to clusters.

Grammar Tester. Purpose of this component is to provide fitness function for entire pipeline with options and parameters configured for every pipeline component and each of its sub-components. There are two ways the fitness function can be evaluated. First, we may use inferred grammar in LG format and try to parse original text with given grammar configured for LG parser. Then, counting percentage of successfully parsed sentences and words per each sentence would give use usability value of the learned grammar, calling it "parse-ability". This approach would work for any language, including unknown languages, however it can not provide warranty that the grammar makes any real sense from linguistic perspective. Second, alternative approach can be tried for languages that are well studied by computational linguistics such as English, where LG dictionaries and grammatical rules are present. For the latter case, we can compare LG parses of original corpus done with native English LG setup as well as with inferred grammar. The proximity across parse trees on sentence-per-sentence basis for these two parses, called "parse-quality", would serve us fitness function rendering to which extent rules that we learn are close to ones created by human computational linguists.

4 Intermediate Results

The results obtained for the current state of the project can be split into a few sections. First, we discuss the use of words, connectors or disjuncts for building the vector space of real-size corpora. Next, we move onto studying different options to count mutual information and perform statistical parsing. Then, we systematically explore the possibility of learning valid Link Grammar dictionaries and rules for simplistic Proof-of-Concept corpora for Turtle and English languages. Finally, we study the possibility of using word-sense disambiguation before parsing, and if its use can improve the quality of Grammar Learning. All corpor and intermediate results are available at http://langlearn.singularitynet.io/data/ site.

At the beginning of the current stage of the project, efforts have been made to try unsupervised word category learning with Gutenberg Adult corpus data, available from the earlier stage of the project [3].

Unfortunately, the data was of low quality, with multiple non-English texts mixed with English ones, lots of special characters and pseudo-graphics included, and only cumulative information on counts of words, connectors and disjuncts extracted from original parses, with no actual parses present. However, using MST parse trees as inputs, we were able to build vector spaces of words and connectors and perform clustering.

It has turned out that, after cleaning the data, for 324 K words, there were 12 M links between the words. Respectively, in vector space of words, there were 324 K original dimensions and in vector space of connectors there were 285 K words on the left and 295 K on the right, so initial dimension for the vector space has effectively doubled. It was found that using original vector space of words, we were not able to identify sensible word categories, due to sparseness of the vector space.

Further, we used a cleaned set of link pairs extracted from the original MST parse trees, accepting only words having more than 85 unique neighbors total, on left and right. It provided us with 31 K words appearing in space of 61 K connectors with 9 M links between the words, so 9.4% words from original data set were supplying 76.2% of links. Using this vector space of connectors, we could clearly identify clusters of words in different languages, proper names, numeric values, and parts of speech. However, due to multiple inconsistencies in training corpus found, we did not proceed with this corpus further for grammar learning, to have re-iteration with this corpus later, when we can confirm that grammar can be actually learned with more simple corpora, as discussed further.

To determine which word-pair counting method better fits the natural "sampling" of parses using LG "ANY" mode, we have computed the Pearson correlation coefficient (PCC) across distributions of FMI values for our POC corpora, between this method and the cartesian combination of per-sentence words, with and without accounting for distance. For the POC-English corpus, we have discovered that the PCC between LG "ANY" and cartesian product, without taking distance into consideration, is 83%; instead, when we consider distance in the cartesian method, we get a PCC of 96%, which indicates that the methods produce very similar FMI. Due to its simplicity and good correlation, we will the default should be method of window-based word co-occurrence counting with account for distance (method b in section Text Parser above) to calculate FMI.

We have also studied to which extent the "expected" English parses, created manually for our POC corpora, correspond to parses provided by Link Grammar Parser with standard English dictionary. It is found that, for the POC-English-NoAmb corpus, the "expected parses" and the LG parses share around 97% of their links. Similarly, for the POC-English-Amb corpus, they share 93% of their links. Based on their similarity, we will use "expected" parses, instead of Link Grammar parses, for "parse-quality" fitness function.

We used "parse-quality" fitness function to compare different versions of MST-Parsing with (a) "expected" parses created manually. For POC-Turtle, it has been found that using (c) "cartesian" combination with account for distance as well as (e) LG "ANY" parses the "parse-quality" is 92% while using (d) "cartesian" combination with no account for distance provides "parse-quality" of 50% only. For POC-English, (c) "cartesian" combination with account for distance provides best quality of 66%, while (e) LG "ANY" parses the "parse-quality" provides 60%, and (d) "cartesian" combination with no account for distance is the worst at 50%. For further work, we choose to use MST-Parsing option (c) "cartesian" combination with account for distance, since it provides the best "parse-quality".

For systematic study of possibility of grammar inference with our pipeline, we have used two simplistic corpora with no ambiguous words in them, namely POC-Turtle and POC-English. For input parses used for grammar learning we used five options: (a) "expected" parses, created manually, with account to LG parse tree conventions; (b) native LG parses with known English LG setup (for POC-English only); (c) Text Parses based on "cartesian" combination of words within window and account for distance; (d) same as (c) but without account of distance; e) LG "ANY" parses, considering all possible parses for the sentence without any grammar knowledge. Four different configurations of Grammar Learner were used. For each of the configurations, different ways of modeling vector space with Connectors or Disjuncts, clustering with Dimension Reduction and K-means (DRK) or collection of Identical Lexical Entries (ILE) and grammar induction with Connectors or Disjuncts were used.

For the two corpora, using different configurations of Parser and Grammar Learner, we were able to get the following results from perspectives of "parse-ability" and "parse-quality". Pearson correlation coefficient between parse-ability and parse-quality has turned to be 85%, which means that **being able to make parse at all means been able to make it right**. When using Turtle language, grammar learning results present 100% parse-ability and parse-quality. When using English - parse-ability in range 50–100% and parse-quality 50–65%. Based on that, we conclude **the problem of automatic learning of formal grammar can be solved with accuracy 50–100%, given corpora that we have tried**. For both corpora, better parse-ability and parse-quality are provided with **MST parsing based on MI with account for distance, building vector space of connectors, then using dimensionality reduction and K-means clustering with subsequent grammar induction by means of either Connectors or Disjuncts**.

5 Conclusion

The primary conclusion of our work is that it is possible to learn formal grammar programmatically based on etalon parses corpus, with possibility to use the grammar for parsing the texts in given language automatically with accuracy in range 50–100%, depending on language. In particular, it has been confirmed for Link Grammar and for very simple controlled corpora in Turtle and English languages.

The secondary conclusion is that statistical parsing can be used for the purpose above, using MST parsing in particular, with account for distance between words when computing mutual information.

Our further plans involve upscaling our approach for using larger corpora, such as Gutenberg Children and Adult an others. We also plan to involve word-sense disambiguation and generalization stages trying to improve parse-ability and parse-quality of the results of parsing with learned grammar. Finally, testing approach would get improved so combination of testing learned grammars on novel corpus data not used for grammar learning will be used for any given language.

References

1. Dupoux, E.: Cognitive science in the era of artificial intelligence: a roadmap for reverse-engineering the infant language-learner. arXiv:1607.08723 [cs.CL] (2018)
2. Goertzel, B., Pennachin, C., Geisweiller, G.: Engineering General Intelligence, Part 2: The CogPrime Architecture for Integrative. Embodied AGI, Atlantis Press, New York (2014)
3. Vepstas L., Goertzel B.: Learning language from a large (unannotated) corpus. arXiv: 1401.3372 [cs.CL], 14 January 2014
4. Sleator D., Temperley D.: Parsing English with a link grammar. In: Third International Workshop on Parsing Technologies (1993)

5. Kolonin A.: Automatic text classification and property extraction. In: 2015 SIBIRCON/SibMedInfo Conference Proceedings, pp. 27–31 (2015). ISBN 987-1-4673-9109-2
6. Barbara, P.: Workshop on Information and Representation Report. In: Workshop on Information and Representation (Washington, DC, March 30-April 1, 1985). Collected Works - Conference Proceedings (021), 151 p. (1985)
7. Friedman, J.: Sheaves on graphs, their homological invariants, and a proof of the Hanna Neumann conjecture. arXiv:1105.0129v2 [math.CO] (2011)
8. Yuret, D.: Discovery of linguistic relations using lexical attraction. arXiv:cmp-lg/9805009 [cs.CL] (1998)
9. Mikolov T., Sutskever I., Chen K., Corrado G., Dean J.: Distributed representations of words and phrases and their compositionality. arXiv:1310.4546v1 [cs.CL] (2013)
10. Cha M., Gwon Y., Kung H.: Language modeling by clustering with word embeddings for text readability assessment. arXiv:1709.01888v1 [cs.CL] (2017)
11. Franco-Penya, H. et al.: An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity. In: 13th International Conference on Machine Learning and Data Mining MLDM 2017, New York, USA (2017)
12. Church, K., Hank, P.: Word association norms, mutual information, and lexicography. Comput. Linguist. Arch. **16**(1), 22–29 (1990)
13. Wall M., Rechtsteiner A., Rocha L.: Singular value decomposition and principal component analysis, arXiv:physics/0208101 (2002)
14. Sculley D.: Web-scale k-means clustering. In: WWW 2010 Proceedings of the 19th International Conference on World Wide Web, pp. 1177–1178, Raleigh, North Carolina, USA (2010)
15. Starczewski, A., Krzyżak, A.: Performance evaluation of the Silhouette index. In: International Conference on Artificial Intelligence and Soft Computing, ICAISC 2015: Artificial Intelligence and Soft Computing, pp. 49–58 (2015)
16. Delpech J.: Unsupervised word sense disambiguation in dynamic semantic spaces arXiv:1802.02605 [cs.CL] (2018)
17. Bartunov, S., Kondrashkin, D., Osokin, A., Vetrov, D.: Breaking sticks and ambiguities with adaptive skip-gram. In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, pp. 130–138 (2016)
18. Hart, D., Goertzel, B.: OpenCog: a software framework for integrative artificial general intelligence. In: Proceedings of the 2008 Conference on Artificial General Intelligence 2008, pp. 468–472. IOS Press Amsterdam (2008). ISBN: 978-1-58603-833-5
19. World Wide Web Consortium: Turtle Terse RDF Triple Language (2012)
20. Gillespie-Lynch, K., Savage-Rumbaugh, S., Lyn, H.: Language learning in non-human primates. In: Brooks, P.J., Kempe, V. (eds.) Encyclopedia of Language Development. SAGE Publications (2014). https://doi.org/10.13140/2.1.3105.3122
21. Patterson, F., Patterson, C., Brentari, D.: Language in child, chimp, and gorilla. Am. Psychol. **42**(3), 270–272 (1987). https://doi.org/10.1037/0003-066x.42.3.270.b