New Trends in Intelligent Software Methodologies, Tools and Techniques
H. Fujita et al. (Eds.)
IOS Press, 2022

doi:10.3233/FAIA220291

Bringing Linguistics to a Programming Class: A Problem of Automatic Text Generation for Describing Data Series

Evgeny PYSHKIN [1] and John BLAKE
University of Aizu, Japan

Abstract. This study focuses on a variety of aspects of teaching programming which can connect the agenda of programming classes to foreign language learning and the broader scope of applied linguistics. In the domain of developing computer-assisted interactive tools for language learning, we can discover a number of interesting and non-trivial problems that can be suggested to students participating in programming classes. Specifically, this study reports an example from our project on the automatic generation of data series trend descriptions to accompany presentation graphs and charts. Originally designed for language learning purposes, this project can be revisited for possible applications as programming exercises that serve as vehicles for a problem-based learning approach to class organization. In general, we believe that through working on, and hopefully, solving practical problems students can apply the content studied in other classes. This application can significantly enhance the knowledge and skills students obtain in programming classes; and thus, improve their professional skill set.

Keywords. programming teaching, problem-based learning, scaffolding, trend description

Introduction

In programming classes, many practical tasks offered to students are designed with the expectation that solutions will use text processing algorithms. The complexity of such tasks ranges from elementary exercises on string processing to intricate problems of lexical and syntactical analysis necessitating the use of statechart models. On the other hand, in applied linguistics there are many tasks that require the development of intelligent approaches to text processing, in connection to both natural language processing and language learning. Language learning-related aspects can be beneficial in creating a programming class teaching environment, where programming exercises can conducted using a problem-based learning (PBL) approach that would favor the tasks appearing in scope of other academic disciplines (e.g. foreign language classes), the latter usually being not considered as directly linked to technological disciplines such as programming or software engineering.

[1]Corresponding Author: Evgeny Pyshkin, University of Aizu, Tsuruga Ikki-machi, Aizu-Wakamatsu 9658580, Japan; E-mail: pyshe@u-aizu.ac.jp

Interestingly, by adopting a PBL approach in which students work together to solve practical problems drawing on their knowledge and experience gained in other software development classes, students significantly enhance their professional skill set, particularly, with respect to software development lifecycle activities, such as subject domain analysis, requirement elicitation, code review, etc.

In this work, we take an example from our project on the automatic generation of data series trend textual descriptions that could accompany graphs and charts presented in a foreign language (in our case – English). This project, originally devised for language learning purposes, can be revisited for possible application to programming classes delivered using a PBL approach. Thus, the original problem is placed into the broader context of development of pedagogic natural language processing software, which requires knowledge from the fields of both digital literacy and applied linguistics. It is worth to mention that PBL models originate in the areas of knowledge, not directly connected to informatics and programming. PBL approaches were adopted widely in medical education [1] and subsequently gained popularity in language education [2,3,4]. However, we definitely observe the suitability of PBL elements to software engineering activities requiring students to work in teams on projects dealing with both authentic and artificial problems. The interdisciplinary nature of solving complex problems not only provides learners with multiple learning opportunities to gain disciplinary knowledge, but also enables them to build transferable skills to help them transition more smoothly from education to employment [5].

1. Background

While writing academic or scientific texts in a foreign language, students often face difficulties in describing charts and data trends adequately even though such descriptions are commonly required in their graduation theses or in written foreign language tests. Descriptions of trends in the result section of student-written research articles and graduation theses tend to be permeated with grammatical and lexical errors [6]. To address this problem, we suggest developing an application that would provide practice opportunities for language learners. Using such an application, the students can either fill in the gaps, complete sentence stems or write the whole text that can be further compared and contrasted against automatically generated model descriptions. To the best of our knowledge, though there are numerous resources that list useful vocabulary and sentences for data series description [7,8] (Figure 1 shows an example), there is no freely-available interactive resource to practice describing such graphs or charts.

The graph	shows	that there has been a	slight steep sharp steady gradual marked	fall rise drop decline increase decrease	in the number of *** since 1981.
Figure 1	reveals				

Figure 1. Substitution sentence for trends (based on *Academic Phrasebank* [8]).

While few researchers are working on data-to-text systems to generate textual summaries (see, for example, [9]), most research in this area focuses rather on the opposite problem of generating graphs from the given textual descriptions [10]. The development of a graph-to-text description generator could be used to raise learner awareness of the suitability of their written descriptions by showing how vocabulary related to trends is used in context. A simple example is shown in Figure 2.

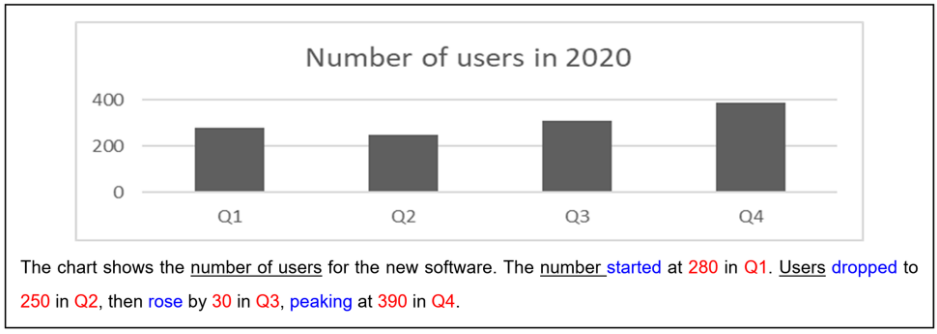

Figure 2. Simple bar chart with colour-coded three-sentence description

The major focus of this paper is neither on improving the ability of students to describe data fluctuation, nor on the specific approach to solve the problem of data trend textual description generation; but on the suitability of using this problem in the context of programming classes. However, in order to demonstrate the feasibility of this task for students, let us share our insights gained through pilot studies on developing an application to generate automated model descriptions of data trends.

Computer science majors taking an elective content and language integrated learning course were set the problem of creating a web application that can visualize and describe trends automatically from a data series. The envisaged users of the web app are Japanese learners of English who need to describe trends in their graduation thesis, but have had limited exposure to and practice at such trend descriptions. In this seven-week course, the instructor reviewed basic string processing operations, introduced the problem and provided students with advice on language analysis.

Students worked in self-managed teams of up to four people to complete the project. The online project management tool, Trello, was used to set milestones and track the progress of the teams. Team leaders were required to submit a requirements analysis, problem breakdown, and pseudocode before beginning the programming phase. Teams submitted draft programs at two intermediate points, or shared access to their chosen repository and collaborative software development platform (e.g. GitHub and GitLab). Feedback on the draft program was provided once by peers and once by the instructor. On completion, students submitted their source code, the URL of their online application and a demonstration video screencasting the more sophisticated functions within their codebase using Terminal.

Axiomatically, the approaches adopted by each team differed, but there were many similarities in terms of both difficulties and how they overcame the difficulties. Although generating the directionality of a trend (e.g. *rise, fall* or *remain stable)* is non-trivial, many groups had difficulty in determining how to select an appropriate adverb of magni-

tude (e.g. *slightly, substantially*). Their initial solutions tended to set absolute (and arbitrary) cut off points based on their initial data series whereas a more appropriate solution would use relative values, since a change of, say 10, may be negligible or substantial depending on the initial data value.

All teams were able to write a program that produced sentences such as: *The number of widgets rose by* 10 *from* 25 *to* 35. However, the repetition of the grammatical subject (*the number of widgets*), the overuse of the same verb (e.g. *rose*) and the inclusion of all three numerical values as prepositional phrases (e.g. change in value, initial value and final value) resulted in very marked (i.e. unnatural and non-human-like) textual descriptions. Another problem was that sentences were demarcated by time period rather than by trend. which means that there may be three consecutive sentences describing a decrease. Some solutions that groups adopted to address these issues included are itemized:

1. *Prepositional phrases* – Randomizing the inclusion of the values so that only one or two of the possible phrases are appended to the string.
2. *Similar trends in consecutive sentences* – When the directionality of the current set of values is the same as the previous set(s) of values, report the change over the whole period in a single sentence instead of multiple sentences.
3. *Repetition of grammatical subject* – When the grammatical subject is a noun phrase (e.g. *the number of widgets*), vary the subject by alternating among the full form, the head of the noun phrase (e.g. *the number*), or replacing with a pronoun (e.g. *it*).
4. *Sentence structure* – Rather than simply using change verbs as the main verb, include some sentences using *be* as the main verb and the trend description as a noun (e.g. *There was a rise...*)
5. *Repetition of verb* – By creating an array of verbs meaning ”go up”, (e.g *rise, climb, increase*), the program can select systematically or randomly from the array when a verb meaning ”go up” is needed.

None of these solutions are particularly challenging for experienced programmers; but for those with little practical programming experience, working out the solution and then creating the program is challenging. The following section deals with the same problem, but rather than being presented as a linguistic problem, it is framed as a vehicle to learning programming.

2. Posing the Problem to a Programming Class

To ensure that problem descriptions follow an appropriate rhetorical and linguistic pattern in a similar manner to the one presented in Figure 2, algorithms are required to be developed to generate the necessary language automatically. This text generation problem is a fruitful niche on which assignments for programming classes may be based. Thus, a simple comparison of adjacent values listed in a time series may be used to determine the direction of the trend (e.g. *increase, decrease* or *remain stable*), though such a determination might not be satisfactory for complex data series like potentially volatile stock rates, technology process measurements, climatic data, etc. Appropriate adjectives and adverbs describing the magnitude of the change can be selected from a vocabulary set based on the relative difference in the values. To avoid excess repetition of the gram-

matical subject, the given noun phase (e.g. the number of users in Figure 2) can be shortened by using the grammatical head (e.g. *the number*) or the tail (e.g. *users*) of the noun phrase, or an anaphoric pronoun (e.g. *it* for the singular noun *number* or *they* for the plural noun *users*). By comparing the difference in the values of the quantity, and taking account of the time period between the values, the *rate of change* (e.g. rapidly, gradually) may also be determined. By combining this information with mix-and-match sentence elements and rhetorical patterns extracted from a corpus, a human-like description can be created using rule-based parsing.

To grade the language level of the generated text to the users, the system needs to take into account both vocabulary and grammatical structures. For a model that classifies users into three language levels (e.g. beginner, intermediate and advanced), the software needs to support several configurations, achieved by drawing upon different lexical sets and combining those lexical sets appropriately. For example, for the beginner level simple sentences (with one main clause) and compound sentences (with two clauses joined by a conjunction) can be generated from a limited set of vocabulary items, while for the advanced level, the full range of sentence constructions and vocabulary can be drawn upon.

Auto-generated descriptions can be transformed into close tests or gap-fill form (similar to the example shown in Figure 3; thus, creating opportunities for language learners to practice. Implementing an algorithm to transform the text into a gap-fill representation can be an excellent derived problem for a programming assignment. Approaches to automatic generation of cloze tests (i.e. fill-in-the-blank texts) has been relatively well developed for different practical areas such as the automatic generation of reading skill exercises [11], language translation exercises [12], history studies [13], and fill-in-the-blank source code for using in programming classes [14]. In the scope of fitting the PBL model to the needs of programming classes, a task of generating such gap-fill structures could be a good motivator for students to learn the approaches similar to those mentioned above; thus, connecting a programming assignment to adjacent domains such as language grammars, graph-based models, *n*-grams, etc. In addition to developing team work, the multidisciplinary nature of PBL approaches enables learners to discover more about related domains in their search for a solution. This search also provides undergraduates with a practical introduction to and experience of conducting research.

Including support for different levels of scaffolding (e.g. at word, phrase, clause or sentence levels) can be an additional requirement for a more challenging programming task. The requirements can also include the possibility to export the resultant text into a format conforming to language learning tool requirements (using XML, for instance).

Gap-fill description	**Full description (level: difficult)**
Norway's krone _____yesterday to its _____ level versus the dollar in more than two years as crude oil _____ for an unprecedented eighth _____month, _____ above $113 a _____. The krone _____ 5.8 percent to 5.2373 against the dollar and _____ yesterday to 5.2304, the strongest level since August 2020.	Norway's krone rose yesterday to its strongest level versus the dollar in more than two years as crude oil rallied for an unprecedented eighth straight month, rising above $113 a barrel. The krone appreciated 5.8 percent to 5.2373 against the dollar and advanced yesterday to 5.2304, the strongest level since August 2020.

Figure 3. Example with gap-fill and full descriptions

Processing a data series for typical trend recognition can be automated by using rule-based or machine learning approaches. Meanwhile, to address the particularities of human perception, interactive tools enabling the marking-up of data series or charts may be beneficial for preliminary data processing, especially with respect to the pedagogic goals of a mixed-level language class, necessitating descriptions at different levels of difficulty. Figure 4 illustrates a possibility to select a number of chart regions that correspond to typical trends, according to expert opinions. The chart shows the rates of gasoline futures over a ten-month period, and demonstrates that practical cases of real-world data are not easy to analyze in a fully automatic mode; decisions on the precise point at which a new trend starts and finishes can be fuzzy. Creating a semi-automated interactive tool to support such decisions can be a good response to fulfill the needs of foreign language learners.

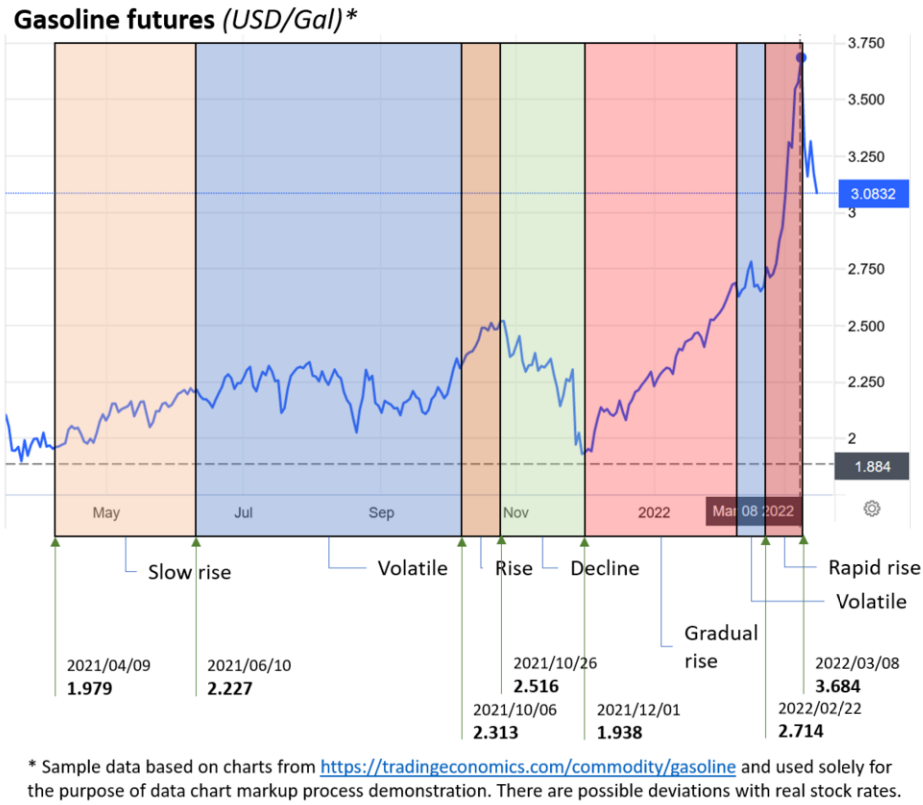

Figure 4. Marking up the chart for further trend description generation

Understanding the vocabulary, or lexicon, is an important point to be taken into the consideration by software creators. Formally, vocabulary V is much wider than simply a bag of words, but a cortege (T_C, T_R, I), where T_C being a partially ordered set of concept types; T_R – a partially ordered set of relation symbols that can be applied to the concepts (types, in software terms) from T_C; I – a set of individual markers (instances, in software terms). According to the theory of conceptual graphs [15], over a vocabulary V, a graph

$G =< C, R >$ can be defined, where C and R state for partitions of concepts and relations respectively. Each concept node C can be described by its concept type and a marker from I, i.e., by some pair $(t,i)|, t \in T_C, i \in I$. Each relation node can be connected to a number of concept nodes. The numbers associated to those connections establish the semantic order of these connections.

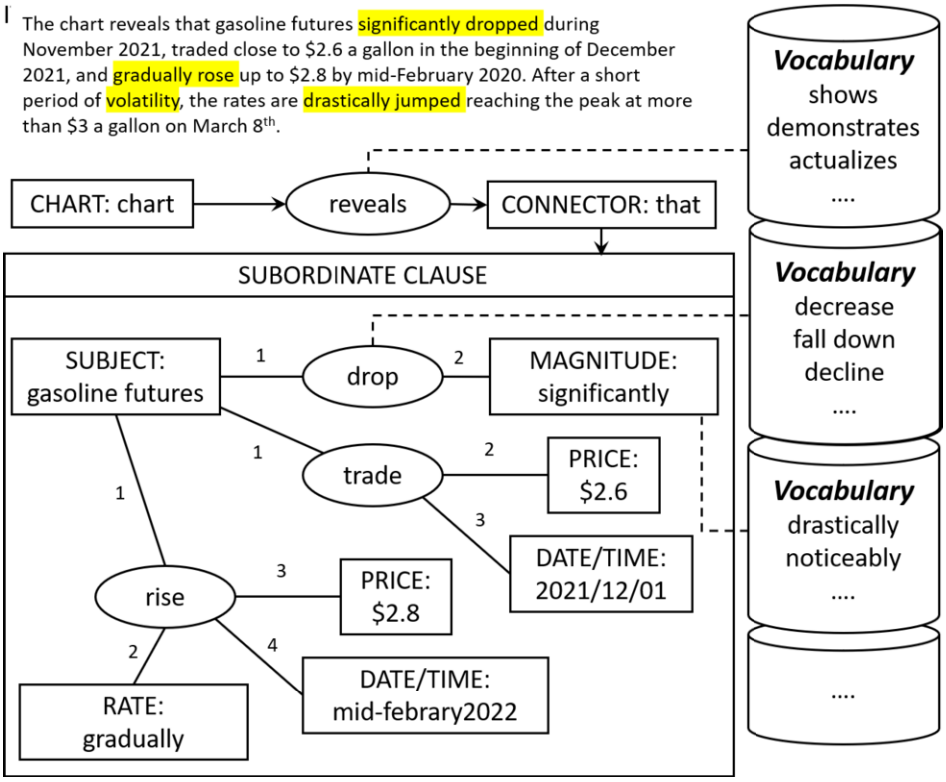

Figure 5. Constructing conceptual graphs for data trend descriptions

Figure 5 illustrates the process of conceptual graph elicitation for the example linked to the sample data as shown in Figure 4. Nested conceptual graphs [16] may be used to represent the complex phrases which include coordinating or subordinating clauses. Though not novel, conceptual graphs still present a suitable formal model for knowledge representation that can not only be used for further descriptive text construction, but also for comparison between different charts and data series. Discovering the similarity of the grammatical patterns used and identifying the semantic constructions to which they refer can be based on conceptual graph homomorphic mapping [17].

3. Academic Outcome

In [18], there is a discussion on project roles that both teachers and students can adopt in a programming class. A role-based approach to class organization is naturally linked to a PBL-based or a flipped classroom approach. Each particular project role is connected

to the particular activities linked to specific software artifacts and refers to particular outcomes and deliverables that are expected to be produced. Figure 6 illustrates a model to instantiate a role-based approach to the case of the project discussed in this paper.

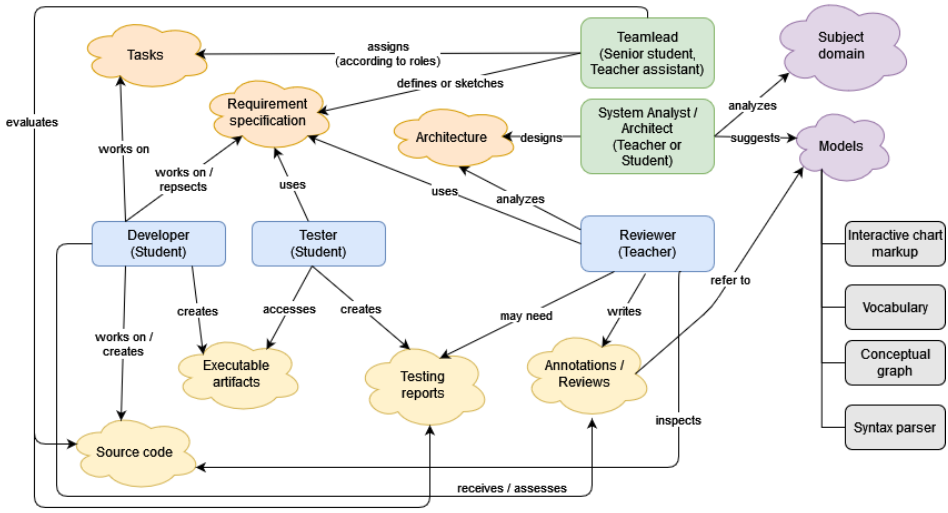

Figure 6. Common-sense ontology representation of role-based workflow in a PBL programming class

The task of generating textual descriptions of graphical or chart data serves as a good example of a multi-aspect exercise problem which requires the application of different independent data models and developer skills, and needs competent coordination between project team members. To sum up, the multi-aspect nature of the problem we investigate requires familiarity with multiple knowledge areas and model domains to be explored by the students under supervision of more experienced instructors including but not limited by the following:

- Approaches to construct good software interfaces linked to a particular subject domain (e.g. interactive interface to mark up the data chart as shown in Figure 4;
- Known knowledge representation models like ontologies or conceptual graphs to be used to generate a formal description of a data trend required for automatic construction of its text version (perhaps, with the added fill-in-the-blank section);
- Language grammar representation models;
- Basic structures used in natural language processing domains; and
- Methodological views on creating software application for learners (e.g. language learners) and their specific properties.

From our teaching experience, we can mention a number of other multi-aspect problems we use in our programming classroom:

- Parsing and circuit implementation of logic expressions;
- Evaluating text document similarity and searching within a collection of text documents using vector space model; and
- Synthesizing simple music melodies based on ABC music notation parsed as introduced in [19].

Though the possible solutions of the above-mentioned problems need students to utilize well-known algorithms and program object organization, they are far from trivial for those beginning their programming journey. The solutions to these problems present a suitable challenge, yet remain feasible with the scope of the limited resources of time and effort in the academic environment of a programming class.

4. Conclusion

From the language learning perspective, the task of generating graph descriptions is one of possible implementations of an idea to visualize basic grammatical language constructions in a way that helps facilitating their comprehension by language learners, especially when learners are able to work interactively with these constructions. Indeed, when grammatical constructions are explained to pupils at high school, typically no formal models of syntax like constituency or dependency grammars are used. Instead, some prototypical structures (or grammatical patterns) may be demonstrated, which is similar to how children acquired their mother tongue with copious amounts of comprehensible input and almost no overt grammatical instruction. Language learners can use patterns explicitly or implicitly: as far as in 1965 McConlogue and Simmons reported that a purely pattern-based English syntax parser was able to show 77% accuracy after experience with only 300 sentences [20]. Nowadays, much higher accuracy of language recognition algorithms can be achieved using machine learning; however, patterns and models still occupy a significant domain of knowledge structuring and representation by humans.

From the programming perspective (where we consider programming as definitely belonging to a class of disciplines related to languages), students' capability to create a model of studied concepts is of crucial importance. As mentioned by Milne and Rowe, the absence of such a mental model is one of usual difficulties in learning programming [21]. Combining foreign language class outcome with the programming assignments can be beneficial for both domains; thus, as mentioned in [22], demonstrating an example of symbiotic relationship connecting language learning to computer technology and software design.

Acknowledgement

This work contributes to the project "Natural language generation of trend descriptions for pedagogic purposes" funded by the Japan Society for the Promotion of Science (JSPS), grant number 22K00792.

References

[1] Wood DF. Problem based learning. British Medical Journal. 2003;326(7384):328-30.
[2] Gallagher SA. Problem-based learning: Where did it come from, what does it do, and where is it going? Journal for the Education of the Gifted. 1997;20(4):332-62.
[3] Learning PB. Speaking of teaching. Center for Teaching and Learning. 2001;11.
[4] Khotimah S. The use of problem based learning to improve students' speaking ability. ELT Forum: Journal of English Language Teaching. 2014;3(1).

[5] Blake J. Real-world simulation: Software development. In: Applied Degree Education and the Future of Work. Springer; 2020. p. 303-17.
[6] Blake J. Corpus-based error detector for Computer Science. In: Proceedings of the Fourth Asia Pacific Corpus Linguistics Conference; 2018. p. 50-4.
[7] Kinnunen T, Leisma H, Machunik M, Kakkonen T, Lebrun JL. SWAN-scientific writing AssistaNt. a tool for helping scholars to write reader-friendly manuscripts. In: Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics; 2012. p. 20-4.
[8] Morley J. Academic phrasebank. Manchester: University of Manchester. 2014.
[9] Jandaghi P, Pujara J. Human-like Time Series Summaries via Trend Utility Estimation. arXiv preprint arXiv:200105665. 2020.
[10] Kandarkar S. Computer Assisted Natural Language Description of Trends and Patterns in Time Series Data [Master thesis]. Technishe Universitaet Muenchen; 2020.
[11] Hill J, Simha R. Automatic generation of context-based fill-in-the-blank exercises using co-occurrence likelihoods and Google n-grams. In: Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications; 2016. p. 23-30.
[12] Panda S, Gomez FP, Flor M, Rozovskaya A. Automatic Generation of Distractors for Fill-in-the-Blank Exercises with Round-Trip Neural Machine Translation. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop; 2022. p. 391-401.
[13] Pannu S, Krishna A, Kumari S, Patra R, Saha SK. Automatic Generation of Fill-in-the-Blank Questions From History Books for School-Level Evaluation. In: Progress in Computing, Analytics and Networking. Springer; 2018. p. 461-9.
[14] Terada K, Watanobe Y. Automatic generation of fill-in-the-blank programming problems. In: 2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). IEEE; 2019. p. 187-93.
[15] Sowa JF. Conceptual graphs. Foundations of Artificial Intelligence. 2008;3:213-37.
[16] Chein M, Mugnier ML, Simonet G. Nested graphs: A graph-based knowledge representation model with FOL semantics. In: KR; 1998. p. 524-35.
[17] Chein M, Mugnier ML. Graph-based knowledge representation: computational foundations of conceptual graphs. Springer Science & Business Media; 2008.
[18] Pyshkin E. On Programming Classes under Constraints of Distant Learning. In: 2020 The 4th International Conference on Software and e-Business; 2020. p. 14-9.
[19] Malan DJ, Malan, Lloyd D, Yu B. Harvard CS50: Introduction to Computer Science. Harvard University; 2019. Available from: `https://pll.harvard.edu/course/cs50-introduction-computer-science`.
[20] McConlogue K, Simmons RF. Analyzing English syntax with a pattern-learning parser. Communications of the ACM. 1965;8(11):687-98.
[21] Milne I, Rowe G. Difficulties in learning and teaching programming—views of students and tutors. Education and Information technologies. 2002;7(1):55-66.
[22] Pyshkin E, Mozgovoy M, Volkov V. Models and metamodels for computer-assisted natural language grammar learning. International Journal of Educational and Pedagogical Sciences. 2015;9(1):60-6.