REFERRING EXPRESSION GENERATION FOR QUESTION ANSWERING AND GRAPH VISUALIZATION[1] [2]

Rygaev I. P. (irygaev@gmail.com)

Laboratory of Computational Linguistics, A. A. Kharkevich Institute for Information Transmission Problems, Russian Academy of Sciences, Moscow, Russia

This paper describes a practical solution for the task of referring expressions generation (REG) in the context of a question-answering system. When an answer to a question is found in the knowledge base the system has to decide how to present the answer to the user, which properties uniquely distinguish the object found from other objects in the knowledge base. Another task where referring expressions would be useful is the semantic graph visualization task. Building on top of the graph-based approach presented by Krahmer et al in 2003 this paper provides some practical improvements to the algorithm, namely: 1) Instead of depth-first graph search we use breadth-first search, which is dramatically faster when a scene graph is big but the description graph to be found is small, 2) Limit on the size (the number of edges) of the resulting description graph to increase performance and avoid useless long descriptions. Also a sketch on linguistic realization of the referring expressions is outlined.

Keywords: natural language generation, referring expression generation, question answering, semantic graph, semantic web, inference

ГЕНЕРАЦИЯ РЕФЕРЕНЦИАЛЬНЫХ ВЫРАЖЕНИЙ ДЛЯ ОТВЕТОВ НА ВОПРОСЫ И ВИЗУАЛИЗАЦИИ ГРАФОВ

Рыгаев И. П. (irygaev@gmail.com)

Лаборатория компьютерной лингвистики Института проблем передачи информации им. А. А. Харкевича РАН, Москва, Россия

[1] This paper presents the results of a joint effort of the team of the Laboratory of Computational Linguistics of the Institute for Information Transmission Problems of the Russian Academy of Sciences. The team includes I. Boguslavsky, L. Iomdin, A. Lazursky, S. Timoshenko, T. Frolova, V. Dikonov, E. Inshakova, V. Sizov and others. I would like to thank my colleagues for their wonderful collaboration.

[2] This work was supported by the RSF grant 16-18-10422, which is gratefully acknowledged.

1. Introduction

The semantic text analyzer SemETAP, under development in the Laboratory of Computational Linguistics of IITP RAS, is aiming at modelling deep understanding of natural language texts (in Russian). The analyzer includes a powerful linguistic processor and various linguistic and extra-linguistic resources—a combinatorial dictionary, an ontology, a repository of individuals, a set of inference rules and an inference engine [Boguslavsky 2011]; [Boguslavsky et al 2010, 2013]. One of the applications of SemETAP is a question-answering system able to answer questions for which there is no direct answer in the original text [Boguslavsky et al 2015]; [Rygaev 2017]. For example, given the sentence:

(1) Зенит не смог спасти матч
Zenit could not save the match

The system can answer:

(2) Кто проиграл?
Who has lost the match?

In order to get the answer the following steps are performed:

1. A language-independent basic semantic structure (BSemS) of the first sentence is built. BSemS consists of a set of binary predicates (RDF triples) and can be seen as a semantic graph where nodes correspond to individuals mentioned in the sentence (including event individuals) and arcs correspond to relations between the individuals.
2. Inference rules are applied to extend BSemS adding new individuals and relations and thus forming an enhanced semantic structure (EnSemS). In our example this step (among other things) adds the knowledge that Zenit has lost the match.
3. BSemS of the question is build. It is similar to that of the affirmative sentence but wh-words are marked in a special way.
4. The question BSemS is used as a pattern to search within the semantic graph of the text. The search returns individuals (graph nodes) corresponding to the wh-words in the question.
5. Along with the node ID certain meaningful information is returned such as the type of the found individual and its name (if exists).
6. Based on this information a linguistic representation of the answer is build and inserted into the text of the question instead of the wh-word, thus generating the answer sentence. Then the answer sentence undergoes some slight modifications (such as agreement and word order change) and is presented to the user. In our example the resulting sentence would be:

(3) Проиграл футбольный клуб «Зенит»
Football club Zenit has lost the match

As mentioned in p. 5 only a few relations (mainly type and name) are currently used to generate the referring expression for the answer. In case the text does not contain the name of the team we are left only with its type (football club) which is not distinguishing enough as can be seen in (4):

(4) Аршавин не смог спасти матч. Кто проиграл?
Arshavin could not save the match. Who has lost the match?

In this case we would like to have the answer *Ashavin's team* or *Arshavin's football club*[3]. The answer *The football club* will not be distinguishing as there are two football clubs in the match. Moreover the type of the potential candidates for the answer is already presupposed by the question (assuming we are talking about a football match). So such an answer does not provide any new information.

The goal of this paper is to present a solution for the general content selection task for referring expression generation (REG). The algorithm should find a minimal distinguishing description of a node in a graph taking into account all existing relations. The second stage (linguistic realization) is beyond the scope of this paper though a sketch of how the problem can be attacked will be outlined.

The paper is organized as follows: Section 2 poses a problem of referring expression generation for question answering, Section 3 presents the graph visualization problem as another task which would benefit from the REG solution, Section 4 discusses related work and existing algorithms for REG including a graph-based approach, Section 5 describes our practical improvements to the graph-based algorithm, Section 6 discusses the evaluation of the new algorithm, Section 7 outlines a sketch of how linguistic realization of the referring expression can be generated, and Section 8 concludes the paper.

2. Referring expressions for answers

Referential choice is known to be a multi-factor probabilistic process [Kibrik et al 2010]. An individual can be referred in discourse by a pronoun, a proper name or a common name potentially modified by an adjective, prepositional phrase or relative clause. Reference also can be of different types such as specific/generic, singular/plural, definite/indefinite and so on.

In what follows we limit ourselves only to specific singular reference. This follows from the nature of the question-answering system. An answer to a question that the system is able to produce is always a specific individual from the knowledge base. In case there are many answers the system will just list them all one by one. Also the usage of pronouns is not an option because the system currently does not take into account the preceding discourse (each question-answer pair is considered to be a separate conversation). Hence ultimately a noun phrase must be generated. But the aim of this paper is limited only to selection of the content for the noun phrase generation.

The problem can be stated as follows:

(5) Given a target node in a semantic graph (called scene graph) find a minimal subgraph (called description graph) which uniquely distinguishes the target node from any other nodes in the graph.

[3] Of course we can look up in the repository of individuals, find out the name of Arshavin's team and use it for the answer. But let's assume the repository of individuals is not available or the team is not there. Anyway, the task of extending the knowledge graph from RI is independent of the task of referring expression generation which is the aim of this paper.

We assume that the information from the description graph will be enough to generate linguistic realization of the referring expression.

Statement (5) covers all our limitations and is generic enough to capture also referring expressions in graph visualization (see the next section). But for question answering certain additional considerations should be taken into account. The content selected for the answer should not include the information which was used to find the node itself. In other word it should not include the presuppositions of the question. Consider the following question-answer pair:

(6) Who won? The winner

This answer is obvious and useless since it is already presupposed by the question itself. To avoid such answers we need to exclude question presuppositions from the scope of search for a description graph. By doing so we need to take into account not only the basic semantic structure of the question but also all the inferences which can be made out of it. Consider the next example:

(7) Who bought the car?
a. The buyer
b. The one who paid
c. The one who received the car

All three answers are useless though only (7a) is contained within the basic semantic representation of the question, and (7b-c) are not contained but rather entailed by it.

So when generating referring expressions for answers we need first to produce an EnSemS of the question (applying inferences to BSemS) and remove the resulting EnSemS from the scene graph before searching for a description graph. In addition to solving the problem with useless answers this scene graph reduction will help with the performance of the REG algorithm.

3. Referring expressions in graph visualization

Another task where short unique referring expressions would be useful is the visualization of a complex semantic graph. In the graphs which are built by SemETAP each node has a unique ID which contains a type of the individual and a certain number postfix. When there are multiple nodes of the same type it is hard to follow which particular individual a node represents.

This is especially problematic for auxiliary nodes such as **`Complete`** or **`EpistModality`**. **`Complete`** nodes are usually generated from perfective aspect of a verb and represent the completion of an event. **`EpistModality`** nodes represent the degree of confidence in a proposition and are attached to any event which is stated or inferred to be a true fact. There can be a lot of auxiliary nodes in the graph and in order to distinguish between them a user needs to check adjacent nodes, sometimes several spans in different directions, which is cumbersome and time-consuming. A unique descriptive expression as a node ID would be really helpful.

See below a graph for a sentence:

(8) Иван купил зонтик у Петра
Ivan bought an umbrella from Peter

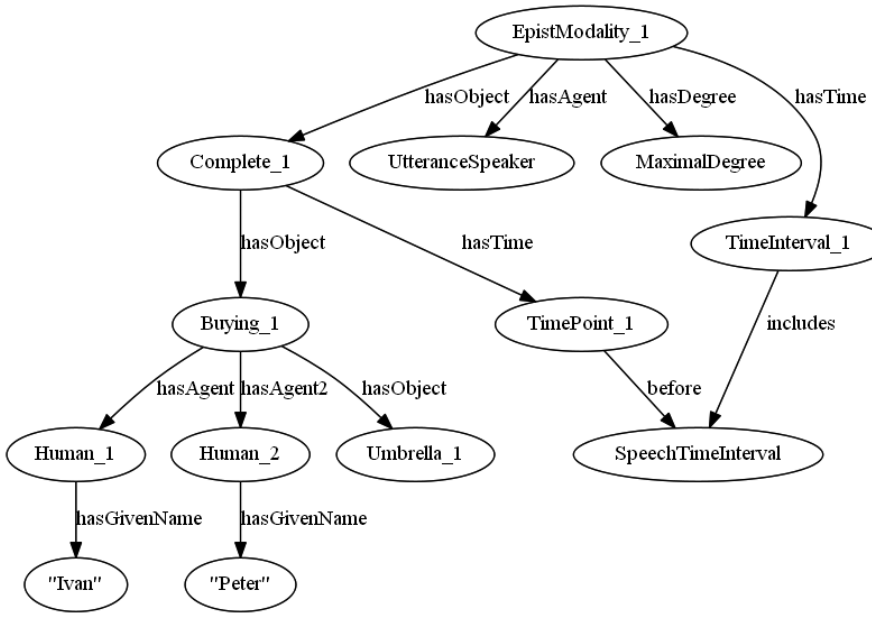

Fig. 1. Basic semantic graph for a sentence
'Ivan bought an umbrella from Peter'

This graph is fairly clear since it is rather small. But when we add inferences to it, the graph becomes unreadable. Instead of providing a picture of it we will list statistics of its node types.

Table 1. Node type statistics of the enhanced semantic graph for the sentence 'Ivan bought an umbrella from Peter'

Grouping	Node type	Number of nodes	Explanation
Objects (4)	Human	2	Two persons—Ivan and Peter
	Umbrella	1	An umbrella
	Currency Measure	1	Money
Events (13)	Buying	1	Ivan bought the umbrella from Peter
	Selling	1	Peter sold the umbrella to Ivan
	Payment	1	Ivan paid for the umbrella to Peter
	Exchange	2	Ivan exchanged the money for the umbrella Peter exchanged the umbrella for the money
	Giving	2	Ivan gave the money to Peter Peter gave the umbrella to Ivan
	Getting	2	Ivan got the umbrella from Peter Peter got the money from Ivan
	Own	4	Ivan owned the money before the purchase Peter owned the umbrella before the purchase Ivan owns the umbrella after the purchase Peter owns the money after the purchase
Auxiliary (42)	Complete	9	Completion for each event except ownership
	EpistModality	22	Facticity of each event and each completion
	TimeInterval	8	Time positions of the events. For some events
	TimePoint	3	they coincide.

For certain nodes (**`Umbrella`**, **`Buying`**, etc.) their class is distinguishing enough, other nodes (people) can be identified by proper names. But when we come to non-unique event types, some descriptive content (such as event arguments) is required to distinguish between them[4]. And for auxiliary nodes we need to include arguments of their arguments as well.

Similar problems arise when one tries to browse open knowledge bases in Semantic Web. If DBpedia (dbpedia.org, [Auer et al 2007]) uses descriptive URIs inherited from Wikipedia article names, Wikidata (www.wikidata.org, [Vrandečić and Krötzsch 2014]) abandons this notation for the sake of multilingualism and uses numeric object IDs such as Q175117. Especially cumbersome is the data structure in BabelNet (babelnet.org, [Navigli and Ponzetto 2012]). The picture below shows how a semantic concept of Apple (fruit) is presented in their linked data interface:

[4] As one may notice node naming is not the only problem in the complex graph visualization. Other issues include proper arrangement of the nodes and the ability for a user to interact with the graph. But even if those two issues are resolved poor node naming will prevent the user from reading and understanding the graph quickly. So we will concentrate on the node naming as this is the only linguistic task in graph visualization.

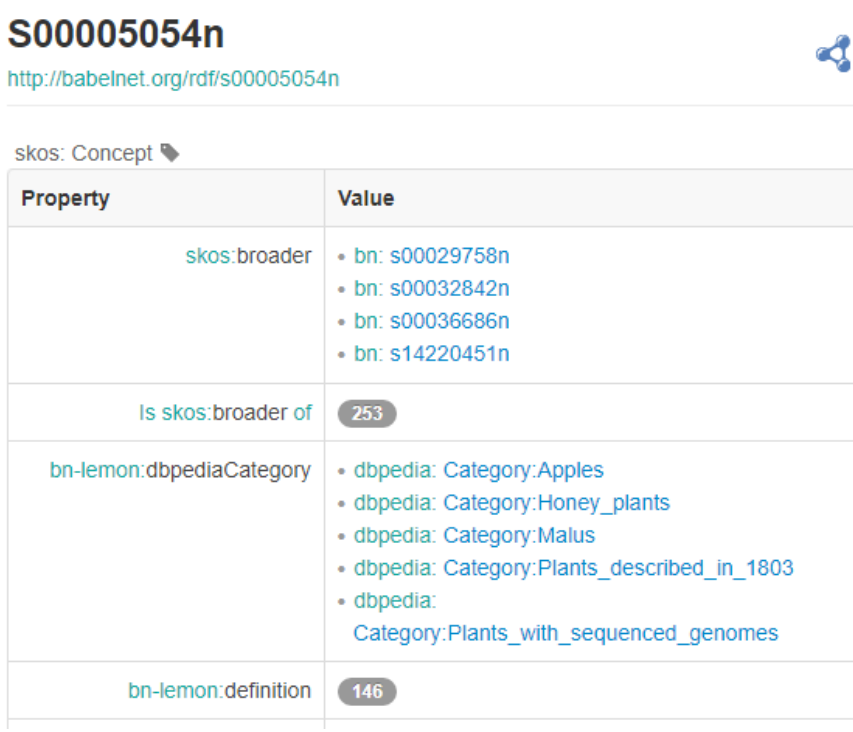

Fig. 2. Semantic concept of Apple (fruit) in the BabalNet linked data interface

If one wants to understand what the broader concepts are, they have to navigate to a particular concept, look at the definition attribute which is also not descriptive but refers to an object named something like s00005054n_Gloss1_EN. And only after navigating to this BabelGloss object one can find a definition of the concept. Adding automatically generated meaningful descriptions instead of numeric concept IDs (or in addition to them) would make the property sheet much clearer.

SPARQL[5] specification [Prud'hommeaux, Seaborne 2008] contains a DESCRIBE query which should return an RDF graph which describes a particular resource (or resources) in an RDF storage, but it is left up to the server to decide which triples to include into the description. This would be another good application for referring expression generation in the context similar to the graph visualization.

4. Existing algorithms for REG

The history of research on the referring expression generation [cf. Krahmer and van Deemter 2012] goes back to [Winograd 1972], who first presented a primitive algorithm for naming objects and events. Since then a number of algorithms have been suggested and evaluated. We briefly discuss the major ones:

[5] A query language for RDF knowledge bases in Semantic Web.

1. *Full Brevity* algorithm [Dale 1989] guarantees to generate the shortest possible distinguishing description. First it tries every single property of the target and checks if it alone rules out all the distractors. If that fails it then tries all possible combinations of *two* properties, then *three* properties and so on until a distinguishing description is found or all the properties are exhausted. This algorithm is computationally expensive and surprisingly of low human-likeness. It was shown that human speakers often produce non-minimal descriptions [Pechmann 1989]; [Engelhardt et al 2006].
2. *Greedy Heuristics* algorithm [Dale 1989, 1992] is more efficient then Full Brevity. It incrementally adds one property to the description—the one which rules out most of the current distractors. Because of its incremental nature (once the property is added it is never removed) it does not always produce the shortest descriptions.
3. *The Incremental Algorithm* [Reiter and Dale 1992] is probably the most influential algorithm in REG. It is similar to Greedy Heuristics but instead of selecting properties based on their discriminating power it uses predefined preference order of the properties. It was shown that speakers prefer certain properties over others when referring to objects [Pechmann 1989]. This algorithm is of polynomial complexity and produces the most natural human-like descriptions. But it requires a preference order to be carefully specified upfront.

It needs to be pointed out that these algorithms originally were tested in a simplified set-up where objects are characterized by their properties only, but not by relations between them [Krahmer and van Deemter 2012: 181]. In our model where almost all properties are in fact relational (even object type is a relation between an individual and a class) this limitation needs to be lifted[6].

There were a number of attempts to adapt the Incremental Algorithm for relational properties [Horacek 1996]; [Krahmer and Theune 2002]; [Kelleher and Kruijff 2006] but unlike simple properties relations do not fit very well into an incremental paradigm. No one would produce a description *'the dog next to the tree in front of the garage'* when *'the dog in front of the garage'* would suffice [Krahmer et al 2003: 57].

[Krahmer et al 2003] suggests a graph-based approach for REG which covers the case of relational properties and fits very well in our knowledge representation framework (since it is already graph-based). They present a branch and bound algorithm [Land and Doig 1960] for finding a relevant subgraph with a cost function to guide the search. Roughly at each step this algorithm enumerates the neighbor edges of the current candidate description graph, checks whether adding an edge will result in a subgraph cost not exceeding the cost of the current best subgraph (if it exists) and if it so checks whether the new candidate rules out all the distractors. If the check is successful then the current best subgraph is updated and the algorithm backtracks, otherwise the same steps are performed on the new candidate.

---

[6] Other limitations such as singular references only, crisp and not vague properties only, ignoring salience in context, etc. [Krahmer and van Deemter 2012: 181] still apply to our work as well. Lifting them is the topic of future research.

Authors argue that their approach (with certain modifications) can mimic the results of all the three algorithms described above. If the cost of adding each edge and each node is the same the algorithm will produce a minimal description as Full Brevity does. If the enumeration of the neighbor edges are performed in a certain order (either based on discriminating power or predefined preference) and the first found description is returned then the results of Greedy Heuristics or the Incremental Algorithm are obtained.

We tried to apply the graph-based algorithm to our tasks. The next section describes some improvements that we had to introduce to it to make the algorithm more practical.

5. Our method

The graph-based algorithm as presented in Krahmer et al 2003:62 is recursive, i.e. it realizes a depth-first search. Starting from one relation the algorithm first explores the whole branch associated with it as far as possible before even trying another single relation. This is not an optimal strategy since we expect a useful referring expression to be relatively short.

Searching for long description (before trying all the shorter ones) can dramatically increase the time required to find a solution if the algorithm happens to take at first the wrong path. Firstly the longer the description (up to a certain threshold) the more its potential for branching, the more new candidates it produces on the next step. And secondly finding distractors for longer descriptions is also more time-consuming. Much more descriptions multiplied by much more time for checking each description makes the algorithm impractical. In our tests the depth-first algorithm could go as far as several dozen relations (still not finding a solution) when a unique description to be found was only several relations long.

To solve this problem we propose a *breadth*-first modification of the graph-based algorithm which (like Full Brevity) first tries every single relation as a full description then every combination of two relations and so on. If there is a relatively short description to be found then the breadth-first search usually finds it much faster than the depth-first search.

But if there is no unique description available then the breadth-first algorithm is slower in arriving at this conclusion. However, to confirm that there is no unique description an algorithm anyway would need to test the longest possible description graph. If the scene graph is connected (and it is usually the case) then the longest possible subgraph would be the whole scene graph. As we mentioned above exploring up to this point is not a practically available option. Also taking into account the following:

1. Long descriptions are not only time-consuming but also not very useful for a user to identify the object.
2. If a unique description does not exist the algorithm still has to return something at least partially useful. It cannot just fail or return an empty string.

We decided to introduce the length limit (in a number of edges) for a description graph. Potential descriptions of the length above the limit are not considered at all.

And when all candidates of maximal length are explored and rejected then the algorithm returns a simple subgraph containing just one edge—a type of the target node. Thus we get acceptable performance for the cost of not always finding unique descriptions (when they are sufficiently long).

In addition to that we realized that forced addition of node types to the description graph is not only beneficial for a user (it produces much more natural descriptions) but also makes the algorithm faster. This is probably due to the fact that our graphs usually contain many edges with the same relation. So a node type plus a relation is much more distinguishing then just a relation. Hence we introduced a rule: whenever a node is added to the description graph its type edge is added automatically as well.

Also we found useful to define a cost function and enumerate neighbors based on the predefined preference order of relations (similar to the Incremental Algorithm). On the top of the preference list we have proper name relations (**`hasName`**, **`hasGivenName`**, etc.) followed by argument relations (**`hasObject`**, **`hasAgent`**, etc.) and so on. This also increases the performance of the algorithm.

6. Evaluation

There are two types of evaluation that can be performed against a REG algorithm—human evaluation of the generated expressions and performance evaluation.

Human evaluation concerns how natural a referring expression is and how helpful it is to identify the target object. Without linguistic realization this type of evaluation cannot be fully performed. But some preliminary validation tests can be made. While surface realization is in progress the resulting expression is presented in a formal language called Etalog which is the language of SemETAP inference rules. It was designed in such a way as to be understandable for linguists without special mathematical or computer-science training. The full description of Etalog is out of scope of this paper. In Fig. 3 we present some examples of Etalog referring expressions in the tree view graph visualization for sentence (4). We hope that they are rather clear and self-explaining.

For evaluation we selected 51 sentences from the corpus of the football high spots (those which were not previously used to developed and test the generation of referring expressions), manually created meaningful questions to these sentences and presented the Etalog answers to four linguists familiar with Etalog asking them to evaluate informativeness and naturalness (human-likeness) of the generated referring expressions. A total of 287 question-answer pairs were evaluated.

Both characteristics were evaluated using a binary scale (yes/no). The informants were instructed to regard an answer as informative if they can unambiguously identify the referred individual within the context of the sentence based on the Etalog expression provided, and the referring expression contains new information (other than what is presupposed by the question itself). And they were instructed to regard the answer as natural if they can imagine someone using such an expression to answer this particular question within the context of this particular sentence.

There were 7 types of referred individuals in the answers. The statistics for each type (as well as the totals) is presented in Table 2. Statistics for persons (football players) is split into two parts: those identified by name and the rest.

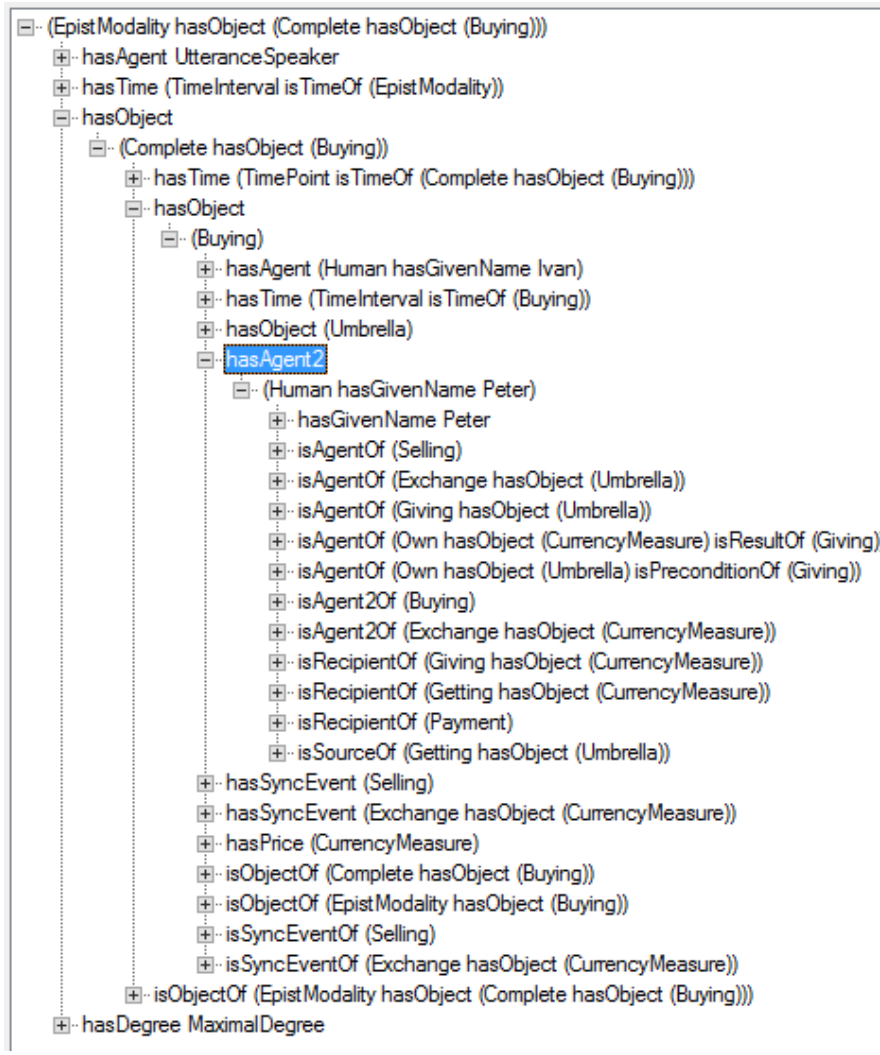

Fig. 3. Tree view of the enhanced semantic graph for the sentence 'Ivan bought an umbrella from Peter'

Table 2. Informativeness and naturalness of the answers by the referred individual type

Referred individual type	Number of answers	Informativeness	Naturalness
Person (identified by name)	81	100.00%	98.42%
Person (identified by other means)	28	36.04%	18.02%
Football team	75	28.16%	17.20%
Place (penalty area, goal area, etc.)	41	62.73%	41.61%
Event (football pass or shot)	32	58.59%	47.66%
Time	22	19.32%	3.41%
Ball	7	100.00%	42.86%
Body part	1	100.00%	50.00%
Total	**287**	**59.27%**	**47.04%**

Table 3 below displays the informative natural, informative unnatural and uninformative examples for each type of referred individuals (where applicable):

Table 3. Good and bad examples for all types of referred individuals

#	Context sentence	Question	Answer
Informative and natural answers:			
(9)	Все тот же Аппаев не попадает даже в створ ворот. All the same Appayev does not even hit the target.	Кто бьёт? Who shoots?	**(Human hasFamilyName "Аппаев")** Appayev
(10)	Думбия и Хонда выводят Мусу к воротам Малафеева, и нигерийцу оставалось лишь не промахнуться. Doumbia and Honda lead Musa to Malafeev's goal, and the Nigerian had only not to miss.	Кому оставалось лишь не промахнуться? Who had only not to miss?	**(Human livesIn Nigeria)** The Nigerian
(11)	После навеса в штрафную в исполнении Кержакова Аршавин блестящим ударом в падении вколачивает мяч в сетку. After a pass by Kerzhakov into the penalty area Arshavin with a brilliant shot in the fall hammers the ball into the net.	За какую команду играет Аршавин? Which team does Arshavin play for?	**(FootballTeam isObjectOf (PlaysFor hasAgent (Human hasName "Кержаков")))** The team which Kerzhakov plays for
(12)	А уже на последней минуте первого тайма Дзагоев не попал в створ ворот из выгодной позиции, пробив рядом со штангой. And in the final minute of the first half Dzagoev missed the target from a vantage point, shooting near the post.	Куда пробил Дзагоев? Where did Dzagoev shoot?	**(Region differentFrom (GoalArea))** Off the goal

#	Context sentence	Question	Answer
(13)	Подача в штрафную Шунина завершается опасным ударом головой Натхо, но голкипер на месте. The feed into Shunin's penalty area ends with a dangerous header by Natkho, but the goalkeeper is at the spot.	Каким ударом завершается подача? Which shot ends the feed?	**(FootballShot hasAgent (Human hasName "HATXO"))** Natkho's shot (The shot that Natkho made)
(14)	На исходе часа игры, Думбия, замкнув прострел Мусы, отправляет второй мяч в сетку ворот Диканя. At the end of an hour of play Doumbia, closing the pass of Musa, sends the second ball into Dikan's goal net.	Когда Думбия отправляет мяч в сетку? When does Doumbia send the ball into the net?	**(TimeInterval finishes (Hour))** At the end of an hour
(15)	Но удар Джуджака оказывается неточным, мяч проходит рядом со штангой. But Dzsudzsak's shot is inaccurate, the ball passes next to the post.	Что проходит рядом со штангой? What passes next to the post?	**(Ball)** The ball
Informative but unnatural answers:			
(16)	Муса навесил в штрафную на Хонду, тот скинул мяч Дзагоеву, который со второй попытки отправляет мяч в сетку ворот Малафеева. Musa lobbed to the penalty area for Honda, who threw the ball to Dzagoev, who at the second attempt sends the ball into Malafeev's goal net.	Кто навесил? Who lobbed (the ball)?	**(Human hasFamilyName "Хонда")** Honda (incorrect answer)
(17)	В следующей атаке хавбек исправился, замкнув в касание передачу Губочана. In the next attack the midfielder corrected himself closing in touch Gubochan's pass.	Кто исправился? Who corrected himself?	**(Human isAgentOf (Attack))** The one who attacked
(18)	Думбия вновь рвется к воротам, но вместо того, чтобы пробить самому, отдает пас на Мусу, которого опережает голкипер. Doumbia runs forth to the goal again, but instead of shooting himself, he passes the ball to Musa, which is left behind by the goalkeeper.	За какую команду играет Муса? Which team does Musa play for?	**(FootballTeam isObjectOf (PlaysFor hasSyncEvent (PlaysFor) hasAgent (Human hasFamilyName "Думбия")))** The team which Doumbia plays for at the same time when someone else plays for another team

#	Context sentence	Question	Answer
(19)	В концовке первого тайма Карсела-Гонсалес упускает очередной шанс своей команды открыть счет в матче, не попав даже в створ ворот. At the end of the first half, Carcela-Gonzalez misses his team's next chance to open the scoring in the match, not even hitting the target.	Куда не попали? What was not hit?	**(GoalArea isTerminalPointOf (GoalEvent))** The goal area where the goal is scored
(20)	Карим Бензема получил пас от Матьё Вальбуена и пробил по воротам, только вот в створ он не попал. Karim Benzema received a pass from Mathieu Valbuena and shot on goal, but he did not hit the target.	Какой сделали удар? Which shot was made?	**(FootballShot hasTerminalPoint (GoalArea isLocationOf (Arriving)))** The shot on the goal where something arrived
(21)	Валладарес переправил мяч в перекладину, от которой тот покинул пределы поля! Valladares repelled the ball into the crossbar, from which it left the field!	Что Валладарес переправил в перекладину? What did Valladares repel into the crossbar?	**(Ball isAgentOf (Leaving))** The leaving ball
(22)	Тем временем Думбия бил головой после навеса Щенникова—неточно. Meanwhile Doumbia shot with his head after Shchennikov's lob—inaccurate.	Чем бил Думбия? With what did Doumbia shoot?	**(Head isInstrumentOf (FootballShot))** The head with which the shot was made
Uninformative answers:			
(23)	Ари выполнял проникающую передачу на Эменике, тот отдал мяч дальше на ход Билялетдинову, и лишь Игнашевич успевает подстраховать голкипера. Ari performed a penetrating pass to Emenike, who gave the ball further to the course of Bilyaletdinov, and only Ignashevich manages to help the goalkeeper.	Кто получил передачу? Who received the pass?	**(Human isAgentOf (Translocation))** Someone who was moving
(24)	В течение минуты Жусилей дважды пытался пробить по воротам Беленова, но оба удара пришлись в защитника. Within a minute, Jucilei twice tried to shot on Belenov's goal, but both shots hit the defender.	По воротам какой команды пытался пробить Жусилей? On which team's goal did Jucilei try to shoot?	**(FootballTeam hasCoach (Human))** The team with a coach

#	Context sentence	Question	Answer
(25)	Ревякин спасает свою команду (сначала) после удара Кержакова, вытащив мяч из-под перекладины, (а затем и Семака). Revyakin saves his team (first) after Kerzhakov's shot, pulling the ball from under the crossbar, (and then after Semak's one).	Откуда вытаскивают мяч? Where the ball is pulled from?	**(Region isObjectOf (Below))** Below something
(26)	Полузащитник "ПСЖ" получил мяч в центре штрафной площади и вторым касанием пульнул по воротам. The midfielder of PSG received the ball in the center of the penalty area and shot on goal with the second touch.	Какой сделали пас? Which pass was made?	**(FootballPass hasLocation (Region))** The pass which is somewhere
(27)	И почти тут же Думбия имел возможность оформить "дубль", но удар у форварда явно не получился. And almost immediately Doumbia had the opportunity to make a double, but the forward's shot was obviously not good enough.	Когда Думбия имел возможность оформить "дубль"? When did Doumbia have the opportunity to make a double?	**(TimeInterval meetsTemporally (TimePoint))** The time right before some point in time

The evaluation shows that the system should be improved in a number of ways. The main problems would be the following:

1. The cost function for the algorithm needs to be configured more carefully. Often the system generated an expression which is formally distinguishing but useless from the human point of view (see examples (17), (25), (27) and others). Probably a more complex cost function is required which takes into account not only the predefined relation order but other things such as types of nodes, population of the required arguments, etc.
2. Duplicated individuals created by different rules are not always combined together by the equality (coreference) rules. This increases the number of distractors and leads to longer unnatural descriptions (see examples (18), (19), (20) and others). The logic to identify and join duplicated individuals should be improved.
3. Concept definitions do not always contain all the necessary information. For example, the definition of the football team should contain the information that it has a coach. If this was included then each football team in EnSemS would have that property and the answer in (24) would not be considered distinguishing.

It should be noted that pp. 2 and 3 above are not related directly to the referring expression generation algorithm but rather to the construction of the scene graph (EnSemS).

For performance evaluation we also present some preliminary figures. They are not final and there is still a potential for optimization. But the tendency is clear—time grows exponentially with the length of the expression. This is the reason we introduced a hard length limit to make the algorithm practically applicable.

Table 4. Average generation time and number of iterations for different description lengths

Description length	Average generation time, ms	Average number of iterations per target	Average time per iteration, ms
1	9.80	1.00	9.80
3	37.40	14.93	2.50
5	369.35	113.96	3.24
7	2,565.00	772.85	3.32

The first column in the table shows the length of the generated referring expression (in the number of edges of the description subgraph). In the majority of cases it is an odd number because edges are usually added in pairs—once a new node is added to the description its type is also added which creates an additional edge in the subgraph. Descriptions of even lengths are generated sometimes too but they do not have enough statistics, so they are omitted from the table.

The second column shows the average time (in milliseconds) required to generate an expression of the given length. The third columns displays the average number of iterations needed, i.e. the number of different descriptions tried before arriving at the solution. And the forth column shows the average time (in milliseconds) of one iteration.

It is clear from the table that the generation time growth mostly comes from the increase of the number of iterations while the average iteration time growth is rather moderate.

7. Linguistic realization

Although a full-fledged linguistic realization or referring expressions is beyond the scope of this paper we briefly present a sketch of how it could be realized.

As mentioned in the previous section we are able to generate referring expressions in Etalog formalism. An Etalog expression can serve as a template for surface realization in a natural language. Consider an example:

(28) `(Own hasObject (CurrencyMeasure) isResultOf (Giving))`

This can be realized in English as follows: *'The ownership of money as a result of a transfer'*. Parallels are straightforward. Roughly what needs to be done is to replace ontological concepts with corresponding words and semantic relations with syntactic ones. This process is exactly opposite to the semantic analysis which SemETAP is already capable of.

One important aspect of an Etalog expression is that it presents a description graph in a tree-like form. This tree can be used as a template for a syntactic tree of the

corresponding linguistic expression. In order to convert an arbitrary connected graph to a tree with a given head the following two steps are performed:

1. Direction of certain edges of the graph is reversed so all edges point from the head to the leaves and not vice versa. This is done through the use of inverse relations. For example, **`isResultOf`** is an inverse relation of **`hasResult`**. Whenever an unwanted incoming relation is found it can be replaced with an outgoing inverse relation.
2. Loops are eliminated. This is done by splitting a node and marking the resulting split nodes with an explicit variable. The second appearance of the variable in the expression lacks any descriptive content and can be realized as a pronoun:

(29) **`(Human ?x isAgentOf (Shaving hasObject ?x))`**
'A person who shaved (himself)'

8. Conclusion

In this paper we presented a practical realization of a graph-based algorithm for the content selection task in the referring expression generation (REG). Starting from the two needful applications of referring expressions—in the question answering and in the node naming for graph visualization, a number of practical improvements for the algorithm were suggested such as breadth-first search (instead of depth-first) and a hard limit for the description length. A preliminary evaluation for the new algorithm was provided and a sketch of the process of generating linguistic expressions based on the formal Etalog expressions was outlined.

References

1. *Auer S., Bizer C., Lehmann J., Kobilarov G., Cyganiak R., Ives Z.* (2007) DBpedia: A Nucleus for a Web of Open Data. Proceedings of ISWC 2007.
2. *Boguslavsky I. M.* (2011). Semantic Analysis Based on Linguistic and Ontological Resources. Proceedings of the 5th International Conference on the Meaning—Text Theory. Barcelona, September 8–9, 2011. Igor Boguslavsky and Leo Wanner (Eds.), p. 25–36.
3. *Boguslavsky I. M., Iomdin L. L., Sizov V. G., Timoshenko S. P.* (2010). Interfacing the Lexicon and the Ontology in a Semantic Analyzer. COLING 2010. Proceedings of the 6th Workshop on Ontologies and Lexical Resources (Ontolex 2010), Beijing, August 2010, p. 67–76.
4. *Boguslavsky I. M., Dikonov V. G., Iomdin L. L., Timoshenko S. P.* (2013). Semantic representation for NL understanding. Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference "Dialogue" (2013), p. 132–144.
5. *Boguslavsky I. M., Dikonov V. G., Iomdin L. L., Lazursky A. V., Sizov V. G., Timoshenko S. P.* (2015). Semantic Analysis and Question Answering: a System Under Development. Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference "Dialogue" (2015), p.62–79.

6. *Dale, R.* (1989) Cooking up referring expressions. Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL), p. 68–75.
7. *Dale, R.* (1992) Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes. The MIT Press, Cambridge, MA.
8. *Engelhardt, P. E., Bailey K. G. D., Ferreira F.* (2006) Do speakers and listeners observe the Gricean Maxim of Quantity? Journal of Memory and Language, 54:554–573.
9. *Horacek, H.* (1996) A new algorithm for generating referring expressions. Proceedings of the 12th European Conference on Artificial Intelligence (ECAI), p. 577–581, Budapest.
10. *Kelleher, J., Kruijff G.-J.* (2006) Incremental generation of spatial referring expressions in situated dialog. Proceedings of the 21st International Conference on Computational Linguistics (COLING) and 44th Annual Meeting of the Association for Computational Linguistics (ACL), p. 1041–1048, Sydney
11. *Kibrik A. A., Dobrov G. B., Zalmanov D. A., Linnik A. S., Lukashevich N. V.* (2010) Referential choice as a multi-factor probabilistic process. Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference "Dialogue" (2010), p. 173–180.
12. *Krahmer, E., Theune M.* (2002) Efficient context-sensitive generation of descriptions in context. Information Sharing: Givenness and Newness in Language Processing. CSLI Publications, Stanford, CA, p. 223–264.
13. *Krahmer, E., van Deemter, K.* (2012). Computational Generation of Referring Expressions: A Survey. Computational Linguistics, 38(1), p. 173–218.
14. *Krahmer, E., van Erk, S., Verleg, A.* (2003). Graph-Based Generation of Referring Expressions. Computational Linguistics, 29(1), p. 53–72.
15. *Land A. H., Doig A. G.* (1960). An automatic method of solving discrete programming problems. Econometrica. 28 (3). p. 497–520.
16. *Navigli R., Ponzetto S.* (2012) BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. Artificial Intelligence, 193, Elsevier, 2012, pp. 217–250.
17. *Pechmann, Th.* (1989) Incremental speech production and referential overspecification. Linguistics, 27:98–110.
18. *Prud'hommeaux E., Seaborne A.* (2008) SPARQL Query Language for RDF. W3C Recommendation 15 January 2008. https://www.w3.org/TR/rdf-sparql-query/
19. *Reiter, E., Dale R.* (1992) A fast algorithm for the generation of referring expressions. Proceedings of the 14th International Conference on Computational Linguistics (COLING), p. 232–238, Nantes.
20. *Rygaev I. P.* (2017) Rule-based Reasoning in Semantic Text Analysis. Proceedings of the Doctoral Consortium, Challenge, Industry Track, Tutorials and Posters @ RuleML+RR 2017 hosted by hosted by International Joint Conference on Rules and Reasoning 2017 (RuleML+RR 2017).
21. *Vrandečić D., Krötzsch M.* (2014) Wikidata: a free collaborative knowledgebase. Communications of the ACM, 2014.
22. *Winograd, T.* (1972). Understanding natural language. Cognitive Psychology, 3(1).