Automatic Comprehension of Geometry Problems using AMR Parser

Anca-Elena Iordan
Department of Computer Science
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
anca.iordan@cs.utcluj.ro

***Abstract*—Automatic comprehension of geometry problems described in natural language is a crucial and challenging stage of numerous automatic geometry problem solvers. These systems should comprehend the existing information in natural language geometry problems with the purpose to extract the geometric relationships between elements and to accomplish automatic solutions using intelligent methods. Abstract Meaning Representation is a popular framework for annotating whole sentence meaning. This paper proposes the addition of a new feature to an existing transition-based AMR parser that constructs AMR graphs from statement of geometry problems described in English language. The new feature consists in explicit embedding of the coreference detection into the parser. Integrating the automatic comprehension method with different geometry systems will greatly enhance the efficiency and intelligence in automatic solving. This approach shows improvements over the best previously published systems for extract information from statement of geometry problems described in English language.**

Keywords - Geometry Problems; Python; Natural Language Processing; AMR Parser

I. Introduction

The Abstract Meaning Representation (abbreviated AMR) [1] is a semantic representation of natural language text, which attempts to catch the connotation of a sentence into a structured AMR graph. The AMR domain retains who accomplishes what to whom in a sentence, and because to this argument, the collected information is kept in a rooted, directed, acyclic graph, appointed the AMR graph [2]. AMR graphs have tags on edges, which are acknowledged as relations, and the leaves of an AMR graph are watched as notions. AMR cannot depict coreferences which cross sentence limits and, likewise, it cannot discern amongst hypothetical or real events. The available mention-pair systems whichever identify coreferences [3] do not analyze entity-level information when establishing if a couple of mentions are referent or not. Therefore, beginning from this ascertainment it was suggest an entity-centric coreference resolution system, which composes coreference chains step by step. This system fits in the pattern stowing type as it engages a pipeline related architecture, where one stage has the grouped mention-pair patterns, which forebodes some rates, and the next stage holds the entity-centric pattern which utilizes previously engendered rates.

A. *Coreference Resolution with Neural Networks*

In the paper [4] the authors introduce a coreference resolution system using a neural network [5], whichever is capable to create distributed appearances of pairs of coreference groups. The appearances are used for the system to train when to merge groups. The system trains when to take alike decisions with the support of a simple first group-ranking procedure. This procedure classifies, in a downward sequence, the rates obtained from the group-ranking pattern with a design to own the highest rate entrant coreference links chiefly, and afterwards selects the ones with the highest rate. The group-ranking system is composed by a single neural network, whose architecture (Fig. 1) has the subsequent subcomponents: mention-ranking pattern, mention-pair encoder, group-pair encoder, and group-ranking pattern.

II. Theoretical Foundation and Analysis of the Existing Systems

A. *Related Work*

Viewed as a closed ensemble, the primary system can be considered as just an application which, plighted as input an English sentence from the statement of a geometry problem [6], releases as output an Abstract Meaning Representation graph (Fig. 2). The available solution can establish only in some measure the wanted output, signifying it can just build an AMR tree (AMR graph without coreferences) for the chosen sentence. Nevertheless, it is not still possible to mark the AMR nodes and relations with labels. Notwithstanding this model can emerge to be rather humdrum in the situation when considered the optimum solutions have under 70.0 out of 100.0 smatch f-scores from a simplistic view [7]. This implies that the AMR area also affords a large possibility for investigation and development.

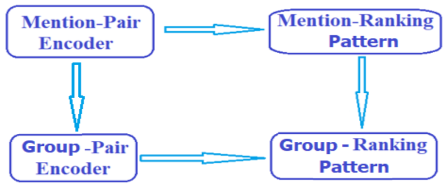

Figure 1. Neural network coreference resolution arhitecture

DOI reference number: 10.18293/SEKE2021-168

Analyzing in profoundness, the primer solution is a transition-based parsing system, whichever enforces a set of transitions to the input sentence with a design to afford its equivalent AMR graph. The transitions appertain to every of the primary states of the system, beginning from the primary state till the final one, are trained with a Stack Long Short-Term Memory neural network [8]. Thereby, utilizing the trained model engendered from the LSTM neural network, the sequence of transitions can be automatically foreshadow starting from a selected geometry sentence. AMR graph [2] is a rooted, directed, acyclic graph, G = (Vx, A), whose edges and leaves are marked. The starting solution transition system includes the fundamental items of an arc-standard parser [9]: a stack (S), a buffer (B) and a group of activities, whichever are applique to the items of the stack and buffer with the purpose to shift by one parser shape to another. A remarkable script involves that the ending state has a blank buffer, a stack holding just one node, which is the root of the AMR graph, and the collection of applique activities on each transition [10]. The parser is trained on data collections that just include English sentences from the statement of a geometry problem and their equivalent Abstract Meaning Representation graphs.

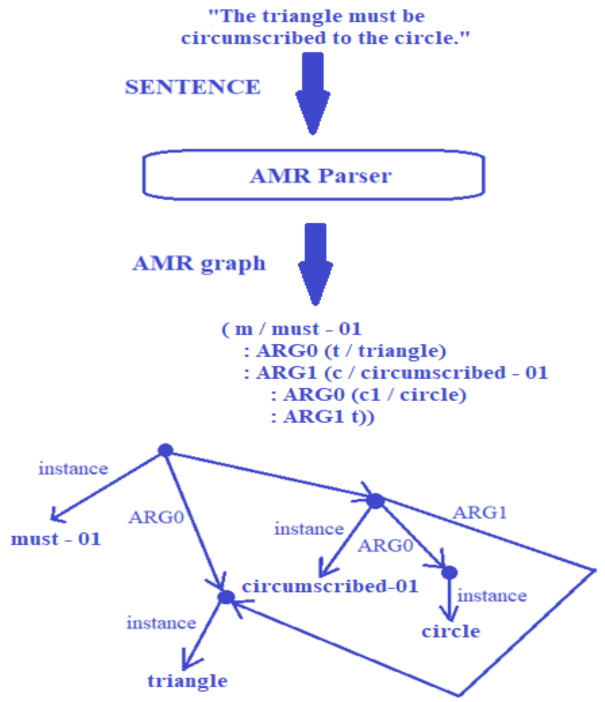

Figure 2. Example of the AMR parser

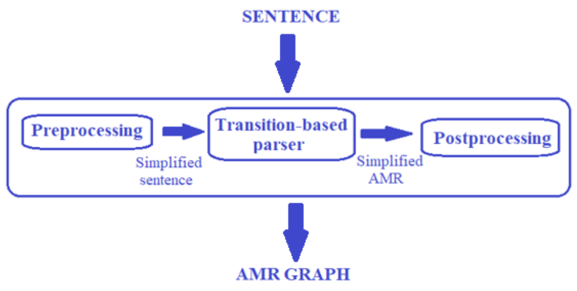

Figure 3. AMR parser structure

B. Preprocessing, Learning, Postprocessing

The primary system utilizes a preprocessing phase to refers several from the attributes of the Abstract Meaning Representation area and create various input models parsed. Similar attributes are presented by Named Entities, Date Entities, quantities, or have-org-roles. The preprocessing phase rental consists in simplification of the learning models with the purpose for the learning method [11] to be qualified to generalize in the best possible way. The learning method underlying on the Stack Long Short-Term Memory architecture, as depicted in [12], and it constitutes the start system's solution, to the tough assignment of transition-based dependency parsing. It utilizes three distinct LSTMs: one for the buffer, which keeps the tokens of the sentence; one to the stack, which includes the AMR nodes; and one for the anterior foreshadowed activities [13]. The postprocessing phase signifies, in a large mode, the opposite of the preprocessing phase, which supplants Named Entities, Date Entities, quantities, have-org-roles, and is constituted from two important activities: subtree reattachment and AMR graph build from the foreshadowed sequence of actions. Furthermore, this restriction is emphasized by the incidence of sentences which include coreferences from our data collections: each data collection comprises approximately 50% of data which includes coreferences [14]. Precise coreferences [15] are referring entities which are contain in the sentence. The precedent selected situations all had clear co-references, from all terms, which referenced a concrete entity, presented in the sentence, and were not deducted.

III. DESIGN AND IMPLEMENTATION

The ascensive data-processing application comprised the next modules: data extraction, preprocessing, action sequence generation, training, postprocessing and evaluation. Data extraction analyzes the input files and takes out the pairs compound from sentence and gold regular AMR graph. The utilized data collections contain files where are data which are split into training, testing and development. They include AMR graphs, their equivalent sentences, and some information such as alignment, record id and annotator [16]. Fully, there are nine collections of data and they were extracted from different sources, such as forums, journals, and geometry books. Compound, they include 18946 items, which are divided amongst the training collection (16104), test collection (1515) and development (1327).

The chosen programming language to develop this system is Python. The main reasons why Python was chosen are accessibility, and scientific and numeric computing. Python is one of the most widely used programming languages for data science and data statistics. Another tool used was spaCy, an open-source library for Natural Language Processing (NLP), written in Python and Cython. It was built to aid in the construction of information extraction or Natural language Understanding systems, or to preprocess text for machine learning and deep learning. NeuralCoref [17] is a neural network pattern for annotating and solving coreference groups and represent an extent for spaCy. It is production-ready and can be enlarged to novel training data collections. NeuralCoref is an adjustment of a mention-pair pattern.

IV. TESTING AND VALIDATION

A. *Evaluation*

The assessment of the system is accomplished on the ensued AMR graphs from the input attempt models. They are resembled towards the Gold regular AMR graphs and a likeness grade is achieved by this procedure. Inasmuch as the AMR area is established on semantic parsing, this implies that exist more AMR graphs which are correlate with a selected sentence. Whereas the interpretations of the AMR graphs are alike, their compositions are, in some extent, various by one to another. Because there is no one exclusive text-to-AMR conversion, the assessment side is sooner complicated. The metric used by the Natural Language Processing collectivity is one which estimates the degree of semantic overlap [17]. The conclusive grade of the system is calculated as an f1 grade over the entities, starting by the foreshadowed collection, which fitted their equivalent gold regular AMR graphs [18].

The f-measure is calculated just once, at the final, afterwards every fit elements and unfit elements were appended simultaneously. The f1 grade is calculated at a great level with the purpose not to distinguish amongst the measures of the AMR graph. Besides from the smatch and f1 grade, it was likewise computed the accuracy of the system as an assessment metric. The accuracy is calculated on the foreshadowed activities reported to the gold regular actions foreshadowed by the Action Sequence Generation pattern. This metric likewise has an analogous deficiency because manifold AMRs can be correlate with an especial sentence. There are distinguished arrangements of the activities which could build the identical AMR graph. Therefore, for assessment, it was utilized the further metrics: accuracy and smatch f-grade. For computing the smatch f-grade were utilized the further three characteristics, foreshadowed by the system starting by the test items: M - the complete number of (parent, relation, child) trios that fitted into the gold regular side of the AMR, T - the complete number of (parent, relation, child) trios foreshadowed through the system and G – the complete number of (parent, relation, child) trios through the gold regular AMR. The accuracy symbolizes the fragment of right activity sequence predictions build from the system towards the gold regular one. If every foreshadowed activity sequence fit in the gold regular activity sequences, the accuracy is 1.0, else, if none of the foreshadowed activity sequences fit the gold regular one, the accuracy is 0.

B. *Results and Comparison with Primary System*

The purpose of this work was to increase the available implementation of the parser. In this scope, it was attempted to manipulate the principal restriction of the existing solution, the reality that the existing system could not operate using coreferences. The collection that the pattern was trained on includes umpteen coreferences and, thanks to this fact, and its other restrictions, the existing system could embrace benefit of just 23.85% of the used information. The boarding utilized was to manipulate coreferences in the preprocessing stage, which would ensue in felling the parser with much divers training items. Thereby, if the parser could be trained on much diverse situations, afterwards, the pattern would be capable to generalize in a superior way and meliorate its capability to foreshadow action sequences. The principal argument wherefore the existing system could parse not many items was since in the collection occur many coreferences. By utilizing the NeuralCoref instrument in this solution, it was identified a fine proportion of the available precise coreferences, attendant in the collection. For every sentence in which precise coreferences were identified, it was enforced to the AMR graph to tree conversion method. The outcomes of this process can be viewed in Table 1. Besides the 4588 sentences in which we established coreferences and enforced the AMR conversion method, the Action Sequence Generation pattern was capable to calculate action sequences for 630 of them, thus, we generated 630 novel parsable items to supply the system. Since the 4588 sentences with their equivalent AMR graphs, 2219 from them were throwed in the preprocessing phase from a certain of the four patterns of the existing solution: nominated entity substitute, data entity substitute or sizes substitute. Accordingly, to the existing 4809 parsable items on which the existing system was entrained, were appended another 630 items, which signified a 13.3% expand in the input information. To calculate the ending outcomes, it was beginning at the existing design of the system and learned it manifold variants by modifying the value of epochs.

The activity sequence size and the word encapsulated scales stand the same from 25 sequence size and from 200 word size. Because the learning side uses so much time consuming, the system was just instructed for next three epoch types: 15, 20 and 25 epochs. In the Table 2 it can notice the outcomes of the system less the coreference manipulation. Established in advance, the metrics utilized to assess the system are smatch f-grade and the accuracy. Resembling the primer outcomes with that which have incorporated the coreference manipulating, it can observe that there is nearly a 5% growing in the smatch f-grade and a 4% growing for the accuracy. It is simple to notice that these ameliorations are firmly bound to the 630 novel training examples generated by the coreference manipulation solution.

TABLE I. OUTCOMES STATISTICS FOR AMR COREFERENCE

Models	**Corefe-rences**	**Identified corefe-rences**	**Prepro-cessing models throwed**	**Primary parsable models**	**Novel parsable models**
926	540	178	153	179	19
1327	886	505	142	202	32
214	78	29	10	84	7
7281	3352	1150	1017	2057	229
689	111	45	11	394	28
6894	3163	2273	714	1445	259
204	140	76	31	22	8
866	484	253	79	216	30
345	86	48	39	156	11
18946	**8930**	**4588**	**2219**	**4809**	**630**

TABLE II. TRAINING OUTCOMES FOR 15, 20 AND 25 EPOCHS

Type	Epochs	Sequen-ce size	Embe-dding size	Smatch f-grade	Accu-racy
Without coreference	15	25	200	38.97	88.99
With coreference	15	25	200	42.87	92.64
Without coreference	20	25	200	39.78	89.76
With coreference	20	25	200	44.36	93.07
Without coreference	25	25	200	40.02	90.02
With coreference	25	25	200	44.97	93.67

Like a remark, the significant distinction among the accuracy and the smatch f-grade it is a consequence of the situation that the mean value for sequence size is 10, thereby, for the network is more convenient to learn from the ultimate side of the action, conversely of learning the complicacy of the sequences of the parsing stages. In this way just grows the accuracy and hold no effect on smatch, because the smatch just handles by AMR graphs.

V. CONCLUSIONS

In this paper it was tried to improve the main limitation of the primary system by coreferences manipulation. This goal has been achieved in the preprocessing and postprocessing stages of the system pipeline since replacing the parser would have implicated beginning a whole novel project from scheme, and not ameliorating the being one. The contribution for ameliorating the system to manipulate coreferences can be divided in two major stages. The first stage has the purpose to discover an algorithm for detection of explicit coreferences in the input sentences. For this purpose, it was used a tool named NeuralCoref, which utilizes a mention pair pattern and two neural networks for approximating grades. Starting from these, two tokens are reported as being coreferent or not. It has identified approximately 50% of the available coreferences from the collections. It deserves emphasized the situation that it was identified just portion of the explicit coreferences existent in a data collection which included both implicit and explicit coreferences.

The succeeding stage was to encapsulate into the AMR parser the reveals from the above stage. In this scope, it was designed a method which would convert an AMR graph within an AMR tree, whensoever precise coreferences were establish in the input sentence. Through this embedding, it was succeeded to engender 630 novel training models to the 4588 being ones, as can be seen in Table 1. This 13.73% growing in entrainment models aided the system to generalize in a better percentage and ensued in a 5% growth of the universal smatch f-grade and accuracy resembled to the primary system outcomes as viewed in Table 2.

REFERENCES

[1] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, U. Hermjakob, and N. Schneider, “Abstract meaning representation for sembanking”, 7th Linguistic Annotation Workshop and Interoperability with Discourse, pp.178-186, 2013.

[2] C. Lyu and I. Titov, “AMR parsing as graph prediction with latent alignment”, 56th Annual Meeting of the Association for Computational Linguistics, 2018.

[3] R. Mishra and T. Gayen, “Automatic Lossless-Summarization of News Articles with Abstract Meaning Representation”, Procedia Computer Science, vol. 135, pp. 178-185, 2018.

[4] K. Clark and C.D. Manning, “Improving coreference resolution by learning entity-level distributed representations”, 54th Annual Meeting of the Association for Computational Linguistics, pp. 643–653, 2016.

[5] I. Muscalagiu, A. Iordan, D.M. Muscalagiu, and M. Panoiu, “The effect of synchronization of agents' execution in randomly generated networks of constraints”, European Computing Conference, pp. 285-290, 2011.

[6] M. Sachan and E. Xing, “Learning to solve geometry problems from natural language demonstrations in textbooks”, 6th Joint Conference on Lexical and Computational Semantics, pp. 251–261, 2017.

[7] J. Buys and P. Blunsom, “Neural AMR parsing with pointer-augmented attention”, 11th International Workshop on Semantic Evaluation, pp. 914–919, 2017.

[8] G.Van Houdt, C. Mosquera, and G. Napoles, “A review on the long short-term memory model”, Artificial Intelligence Review, vol. 53, pp. 5929–5955, 2020.

[9] Y. Liu, W. Che, B. Zheng, B. Qin, and T. Liu, “An AMR aligner turned by transition-based parser”, Conference on Empirical Methods in Natural Language Processing, pp. 2422-2430, 2018.

[10] I. Muscalagiu, H.E. Popa, and V. Negru, “Improving the performaces of asynchronous search algorithms in scale-free networks using the nogood processor technique”, Computing and Informatics, vol. 34, pp. 254-274, 2015.

[11] A. Iordan, M. Panoiu, I. Baciu, and C.D. Cuntan, “Design of an intelligent system for the automatic demonstration of geometry theorems”, International Conference on Telecommunications and Informatics, pp. 221-226, 2010.

[12] C. Wei, “Development of Stacked Long Short-Term Memory Neural Networks with Numerical Solutions for Wind Velocity Predictions”, Advances in Meteorology, vol. 2020, article number 5462040, 2020.

[13] M. Damonte, S. B. Cohen, and G. Satta, “An incremental parser for abstract meaning representation”, 15th Conference of the European Chapter of the Association for Computational Linguistics, pp. 536–546, 2017.

[14] C. Wang and N. Xue, “Boosting transition-based AMR parsing with refined actions and auxiliary analyzers”, 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 857–862, 2015.

[15] K. Werling, G. Angeli, and C. D. Manning, “Robust subgraph generation improves abstract meaning representation parsing”, 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 982–991, 2015.

[16] L. Nguyen, V. Pham, H. Minh, D. Dinh, and T. Manh, “Integrating amr semantic graphs to convolutional neural machine translation“, ICIC Express Letters, vol. 12, pp. 133-141, 2021.

[17] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, “End-to-end neural coreference resolution”, Conference on Empirical Methods in Natural Language Processing, pp. 188–197, 2017.

[18] N. Stylianou and I. Vlahavas, “A neural Entity Coreference Resolution review”, Expert Systems with Applications, vol. 168, article number 114466, 2021.