Contribution ID: **222** Type: **Sectional reports**

Botnet in PyPy to speed up the work of the Earley parser

Thursday 13 September 2018 15:45 (15 minutes)

Extraction of information from texts is a crucial task in the area of Natural Language Processing. It includes such tasks as named-entity recognition, relationship extraction, coreference resolution, etc. These problems are being resolved using two approaches. The first one is the rules-based approach and the second one is machine learning. Solutions based on machine learning are currently very popular but work well only with frequently used entities such as person, company, or organization. They require the presence of a large tagged dataset. With attributes from narrow subject areas and facts, machine learning works much worse. It is better to use this approach in writing rules for context-free grammars. The problem here is that the more grammars there are, the slower the analyzer works. Often the speed of the algorithm is less than 1 KB of text per second. For my project concerned with collecting information and statistics in the subject area of oil and gas geology, I created a system that includes a botnet using the Selenium library, in which one computer generates search queries from a list of objects and collects the resulting links. Then, the resulting links send the tasks to other computers via the REST service based on the asynchronous queue implemented in the Flask framework. To avoid link duplication, hashing was used with Redis. Next, the task is occupied by a botnet consisting of various computers. They take the task and depending on the content of the link carry out the following: if this is an ordinary site, then its contents are parsed; if this is a doc/docx/pdf document, then it is downloaded and then text is extracted from it using the textract library. After saving the text, algorithms are used to extract entities and thematic attributes using the GRL parser on context-free grammars. The first step to accelerate is the use of distributed computing as described above. The second step in the acceleration that was undertaken in this study is the use of the PyPy interpreter for Python, which compiles the code in the C language. It accelerated the work of the algorithm 4 times on average, but the consumption of RAM increased by ~ 25%. The calculations involved 8 computers with 4 threads each. Thus, the use of distributed computations together with the replacement of the standard Python interpreter with PyPy allowed to increase the speed of the extraction of facts increased ~ 128 times.

Summary

The use of distributed computations together with the replacement of the standard Python interpreter with PyPy allowed to increase the speed of the extraction of facts increased ~ 128 times using GRL parser on context-free grammars.

Authors: Mr KULNEVICH, Aleksey (Dmitrievich); Mr RADISHEVSKIY, Vladislav (Leonidovich)

Presenter: Mr RADISHEVSKIY, Vladislav (Leonidovich)

Session Classification: 11. Big data Analytics, Machine learning

Track Classification: 11. Big data Analytics, Machine learning