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

Radishesvkii Vladislav
Kulnevich Aleksei

Dubna, 13 september 2018

Introduction

Natural Language Processing

is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data.

Aim: To build intelligent computers that can interact with human being like a human being

Why NLP?

- Huge amount of data
  - Internet = 1.9 billions websites, 7.6 billion users (https://www.internetworldstats.com/)
  - Text data – web sites, blogs, tweets, social networks, ...
  - Audio data – speech, ...
- Applications for processing large amounts of texts require NLP expertise

Introduction

Natural Language Processing

Use cases:

- *Politics*
- *Sentimental analysis (brand & feedback analysis)*
- *Cognitive search (on the semantic content)*
- *Question-answer systems (chatbots)*
- *Speech recognition & generation*
- *Machine translation*
- *. . .*

Objectives of work

Collect text data

Receiving text data from documents of various formats
Search for websites with the right information and extract text from them

Extract entities & key-values

Retrieve indications to named entities and domain attributes

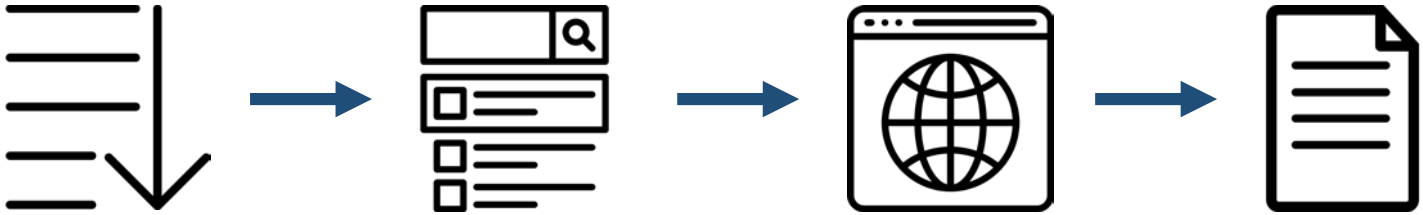
Speed up the information retrieval process

Use multiprocessing and distributed computing

Collect text data

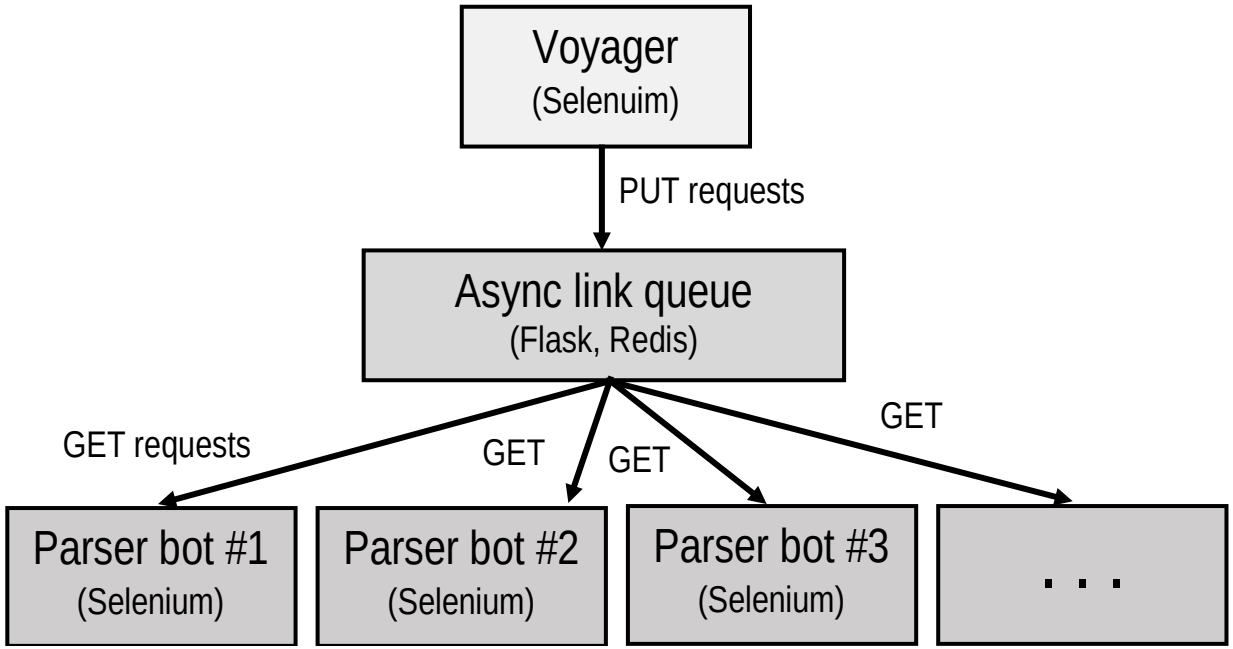
Data collection scheme

Generate queries → *Get search results* → *Open page or download file* → *Extract text & save*



Collect text data

Distributed page parsing



Test Automation
Selenium

Flask
web development, one drop at a time

redis

Extract entities & key-values

Earley parser *is an algorithm for parsing strings that belong to a given context-free language.*

The Earley parser executes in cubic time in the general case $O(n^3)$, where n is the length of the parsed string.

It was first introduced by Jay Earley in 1968

Extract entities & key-values

Earley parser (implementation) - Yargy

natasha / yargy

Watch 13 | Star 79 | Fork 14

Code | Issues 9 | Pull requests 0 | Projects 0 | Wiki | Insights

Tiny package for information extraction

morphology | information-extraction | russian-specific | earley-parser

337 commits | 1 branch | 0 releases | 4 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

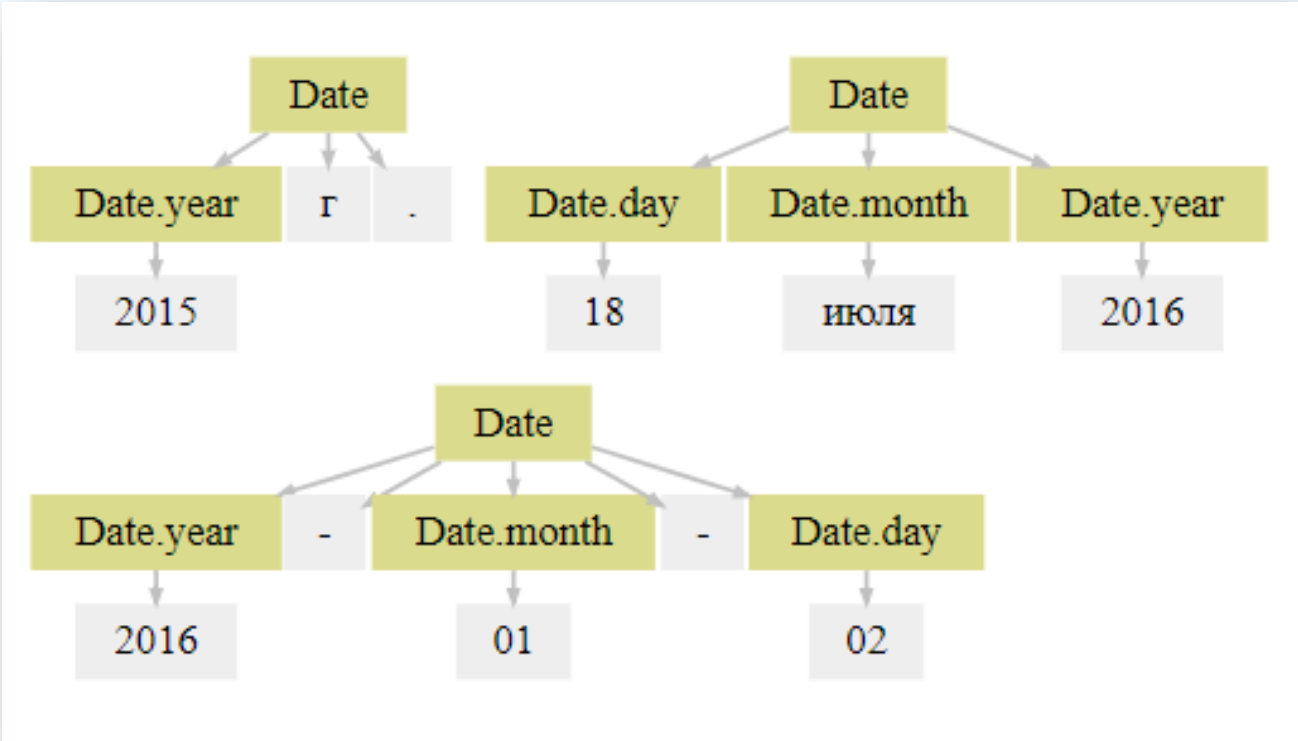
takiholadi and dveselov Fix typos in docs via spell-check (#52) — Latest commit e79002d 11 days ago

docs	Fix typos in docs via spell-check (#52)	11 days ago
yargy	Fix py2	2 months ago
.gitignore	Initial commit	2 years ago
.travis.yml	Run predicate doctests	a year ago
CHANGELOG.md	Up readme	6 months ago
LICENSE	Initial commit	2 years ago
README.md	Up badges	6 months ago
appveyor.yml	Update appveyor.yml	a year ago
requirements.txt	Do not currently use jellyfish	8 months ago
setup.py	Bump version	2 months ago
tox.ini	Support recursive grammar	a year ago

https://github.com/natasha/yargy

Extract entities & key-values

Grammar – date extraction example



Extract entities & key-values

Grammar - constructions

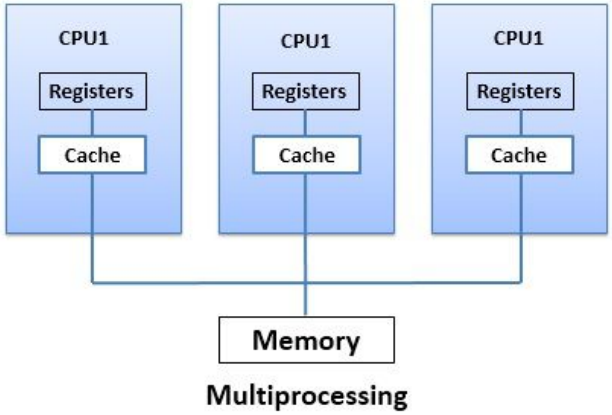
- ***Tokenization*** *is the process of demarcating and possibly classifying sections of a string of input characters*
- ***POS tagging*** *- define the part of the speech of a word, its genus, number, case*
- ***Predicates*** *–base constructions for writing grammars (and, or, caseless, eq etc.)*
- ***Relation*** *- gender_relation, number_relation, case_relation, gnc_relation*
- ***Gazetteer*** *is a dictionary. In it, you can define a set of words suitable for a description, and use them in grammar*

Speed up the information retrieval process

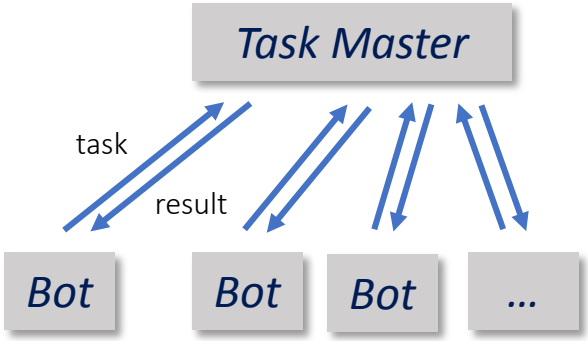
processing speed is very slow ~ 1500 bytes / sec

Multiprocessing & distributed computing

- Multiprocessing



- *Distributed computing*



The calculations involved **8 computers** with **4 threads** each (*x32*)

Speed up the information retrieval process

Python -> PyPy

***PyPy** is a fast, compliant alternative implementation of the Python language*

For our grammars, he accelerated the processing speed of the text by an average of ***4 times,*** but the consumption of RAM increased by ***~25%***

Conclusion

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 **~ 4 times** using Earley parser on context-free grammars. Multiprocessing & distributed computing (8 computers & 4 threads) allowed to speed up also in 32 times.

Thank you for attention!