Performance and load testing results of WebSpellChecker v5.6.2

This document describes the results of performance tests for WebSpellChecker Web API. The performance and load were tested on the following setup:

- 100 users accessing the server simultaneously;
- Cache enabled for spell checking purposes;
- Certain hardware and software used (EC2 m5.large instance with 2 CPU and 8 GB RAM);
- Number of words to be checked (1K words or 6K characters);
- Number of spelling and grammar problems in the text (15 grammar problems | 35 misspellings), around 5% of errors;
- The type of language used for checking (17 default languages).

Testing goal

Our main goal was to observe the **response time** of text processing and **CPU utilization** on the server when 100 users send simultaneous requests on various languages to the server with WebSpellChecker v5.6.2.

Environment and testing tool

- We used Apache Jmeter 5.1.1 as a performance measuring tool.
- The machine used was AWS EC2 m5.large instance with 2 CPU and 8 GB RAM.
- The version of the WebSpellChecker Server package is 5.6.2 (released on July 8, 2020).

Testing process

We have run our tests continuously for each of the languages in the default language group (17 languages). The cache setting was enabled for all sets of tests. The following combinations of text to be checked are used for each language:

- 1K words with 35 misspellings only;
- 1K words with 15 grammar problems only;
- 1K words with 15 grammar problems and 35 misspellings.

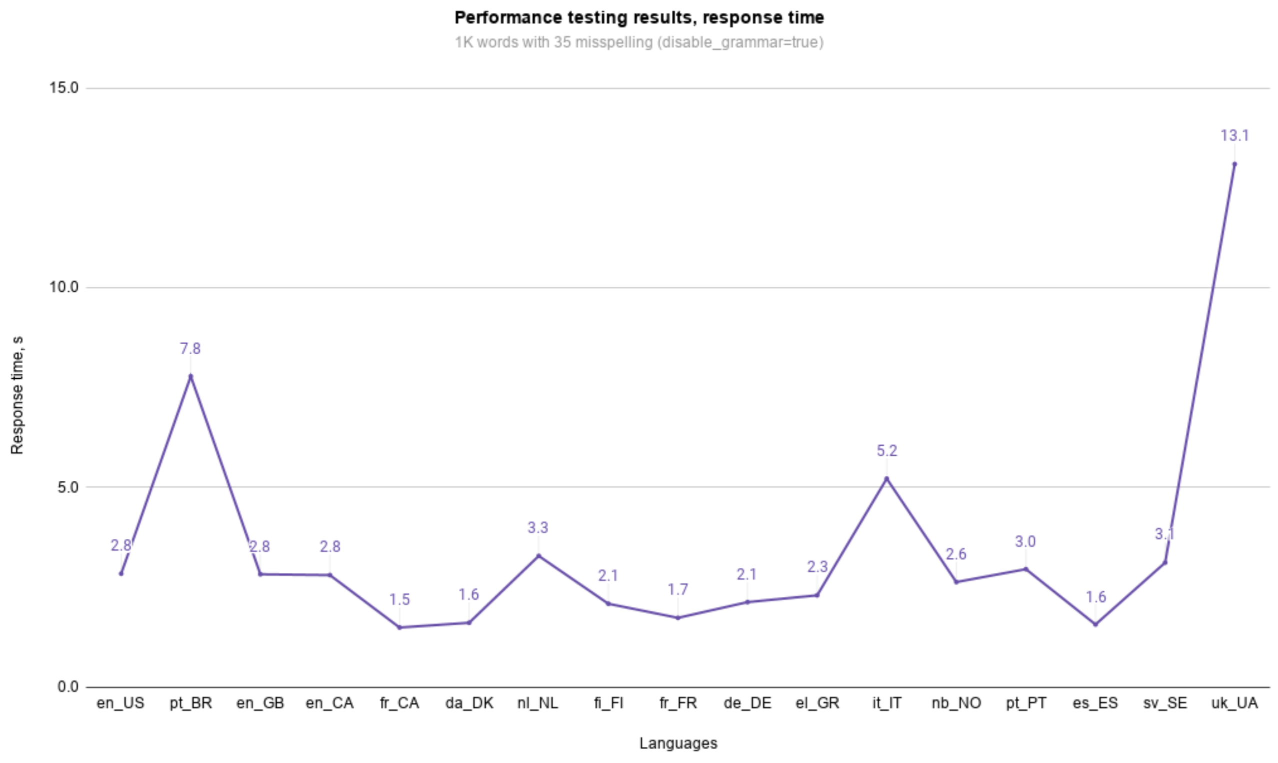
The measured metrics were the response time and CPU utilization.

Observations

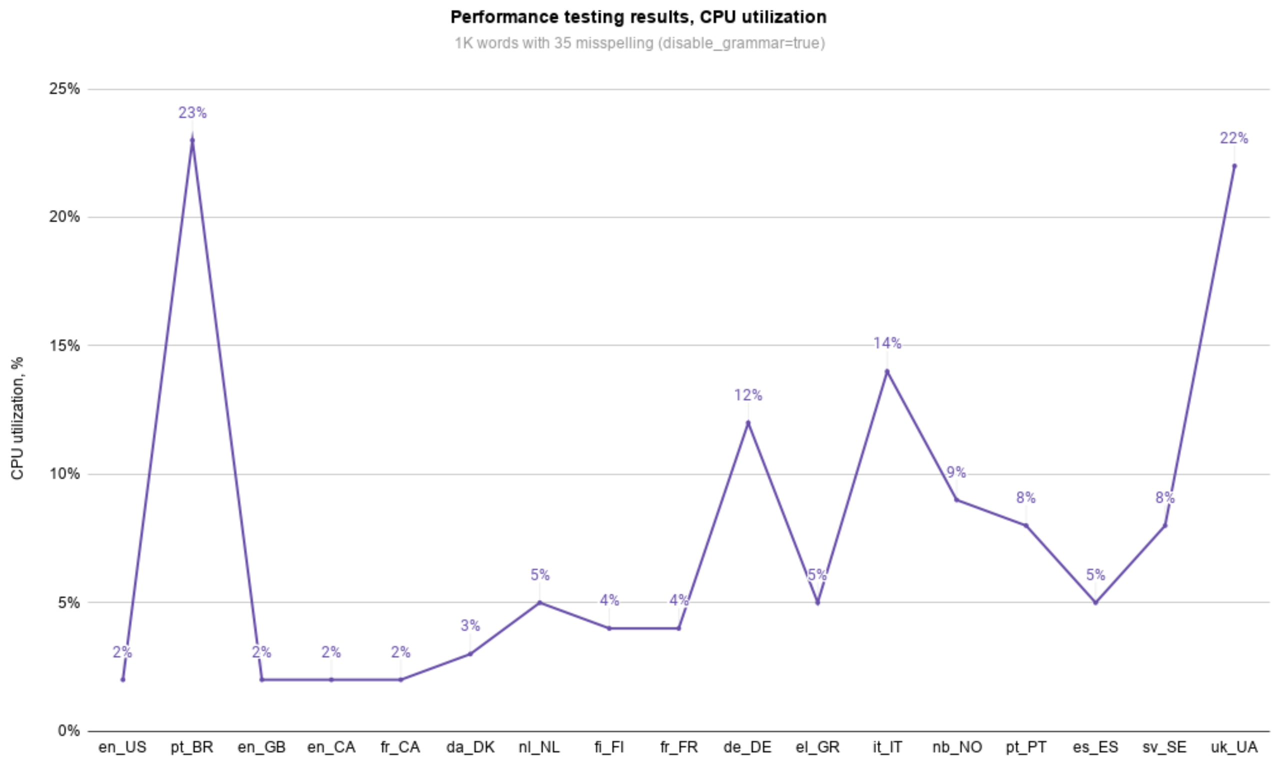
Our observations are presented in the charts below.

Response time and CPU utilization (only 35 misspellings)

The chart below represents **response time** results aggregated by language when there are **only 35 misspellings** in text of 1K word size.

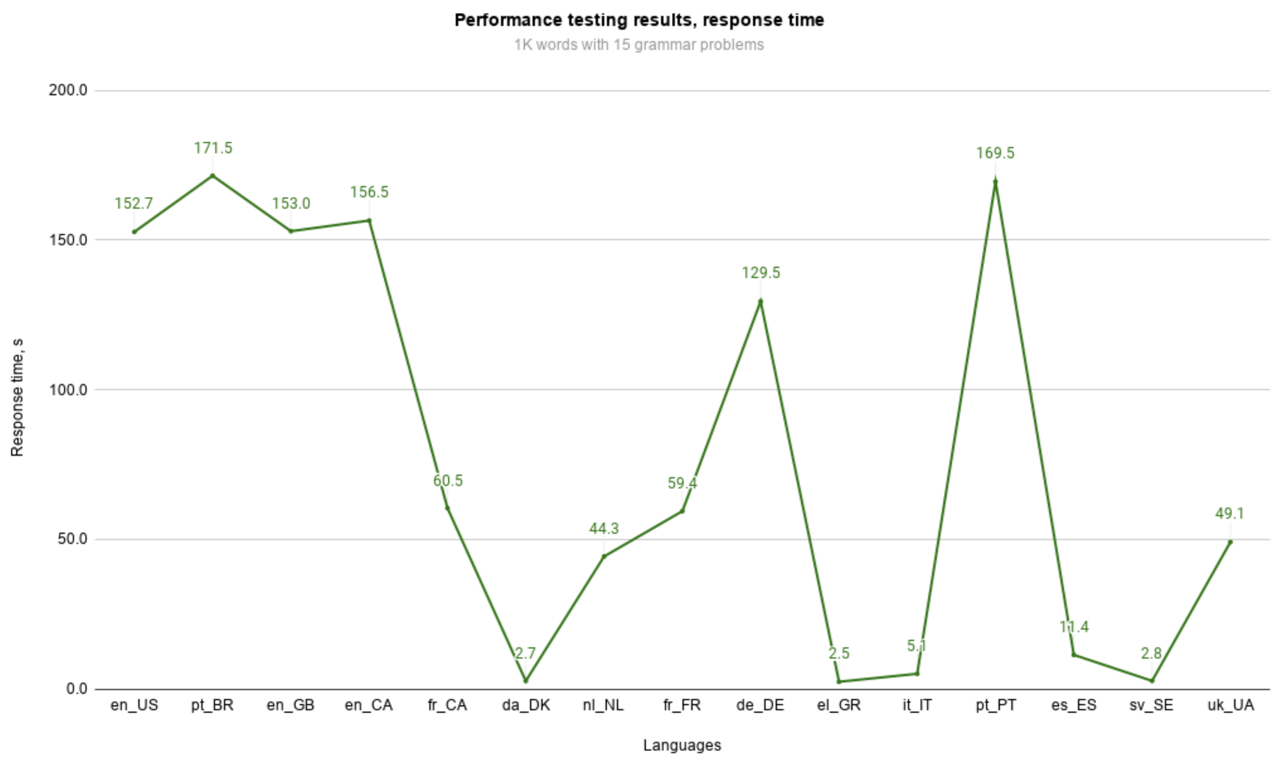


The chart below represents **CPU utilization** results aggregated by language and user tiers when there are **only 35 misspellings** in text of 1K word size.

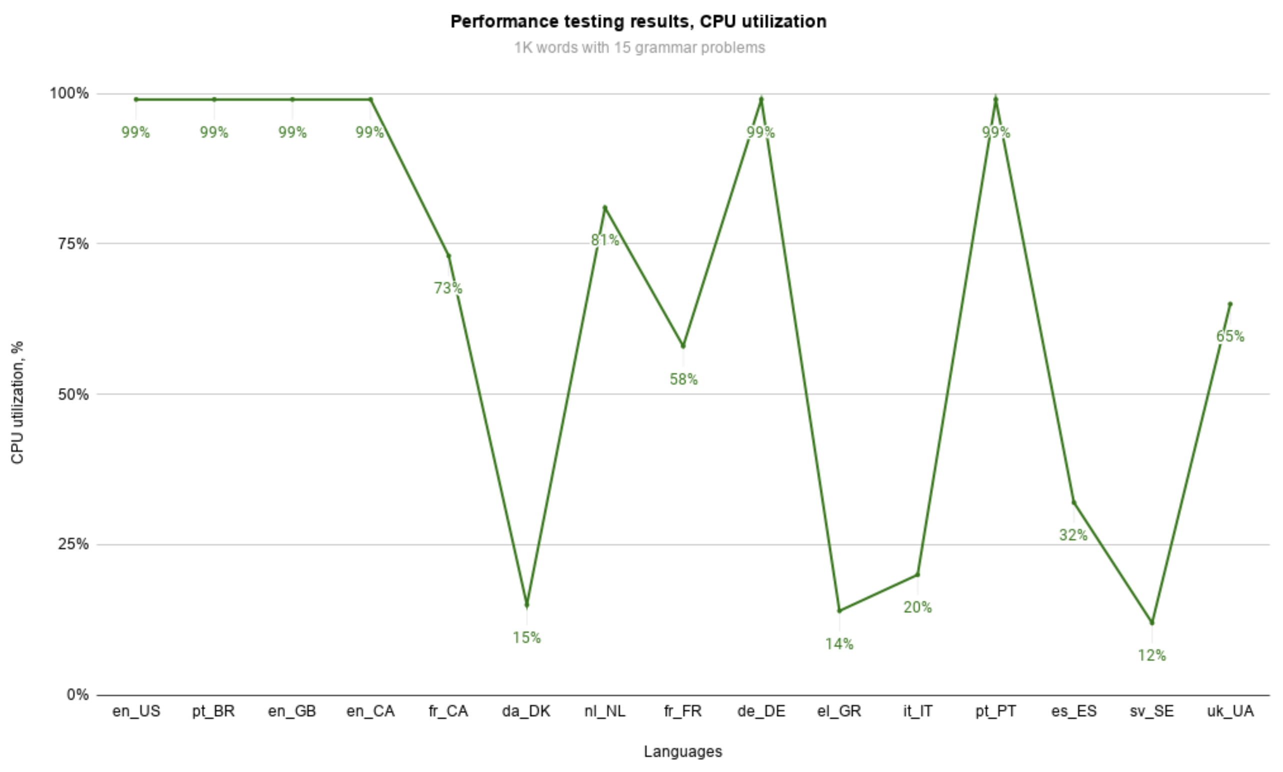


Response time and CPU utilization (only 15 grammar problems)

The chart below represents **response time** results aggregated by language when there are **only 15 grammar problems** in text of 1K word size.

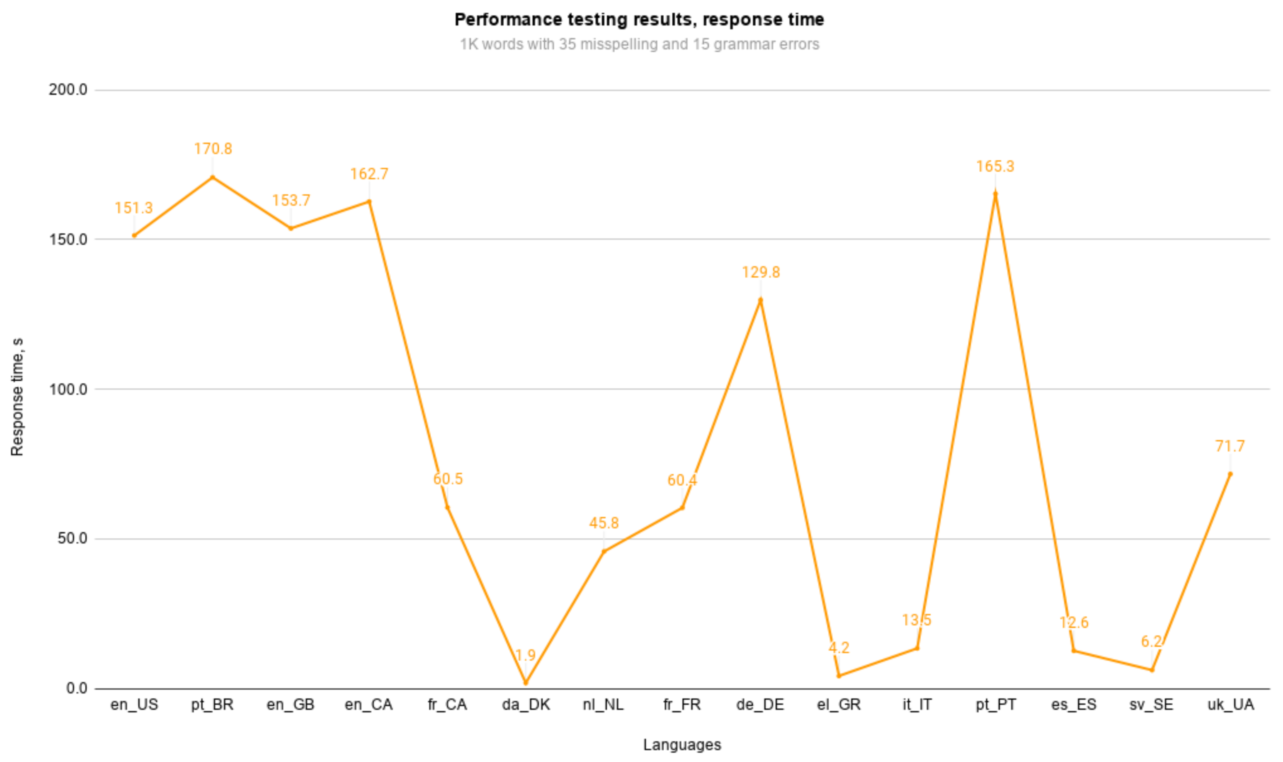


The chart below represents **CPU utilization** results aggregated by language when there are **only 15 grammar problems** in text of 1K word size.

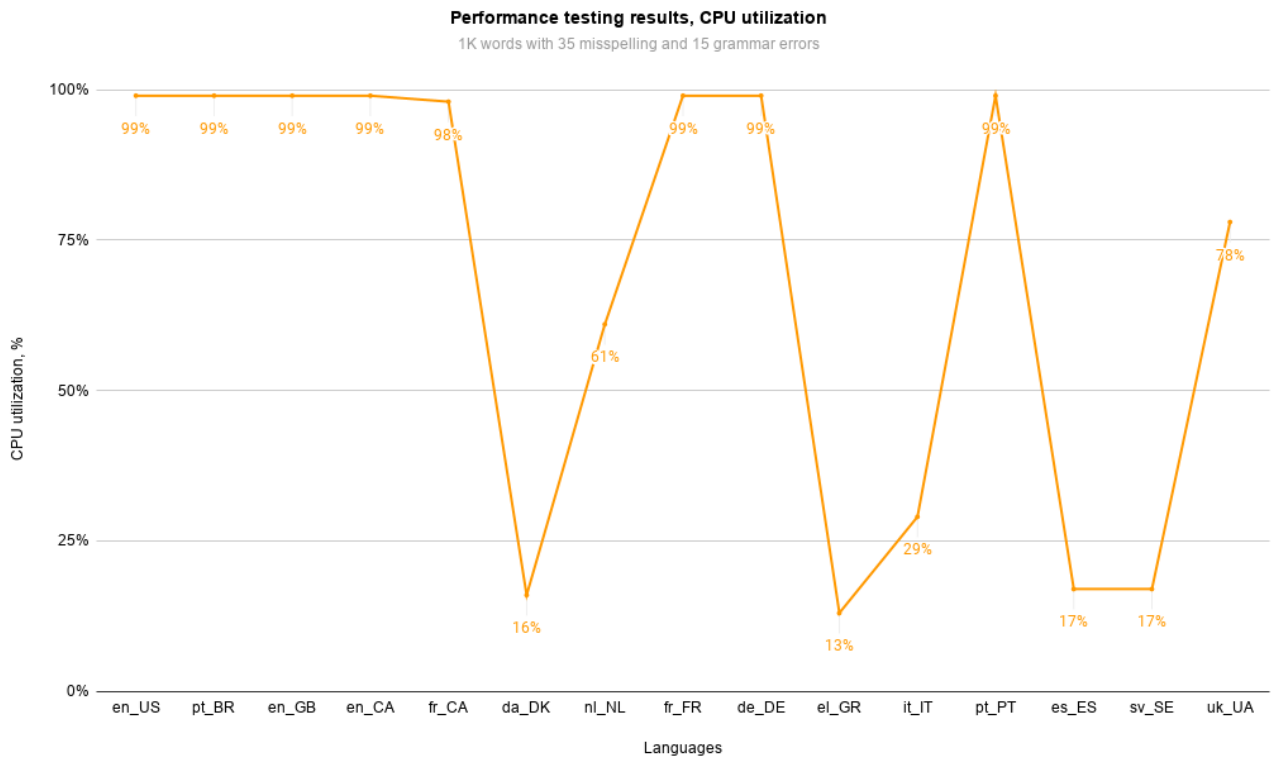


Response time and CPU utilization (35 misspellings and 15 grammar problems)

The chart below represents **response time** results aggregated by language when there are **35 misspellings** and **15 grammar problems** in text of 1K word size.

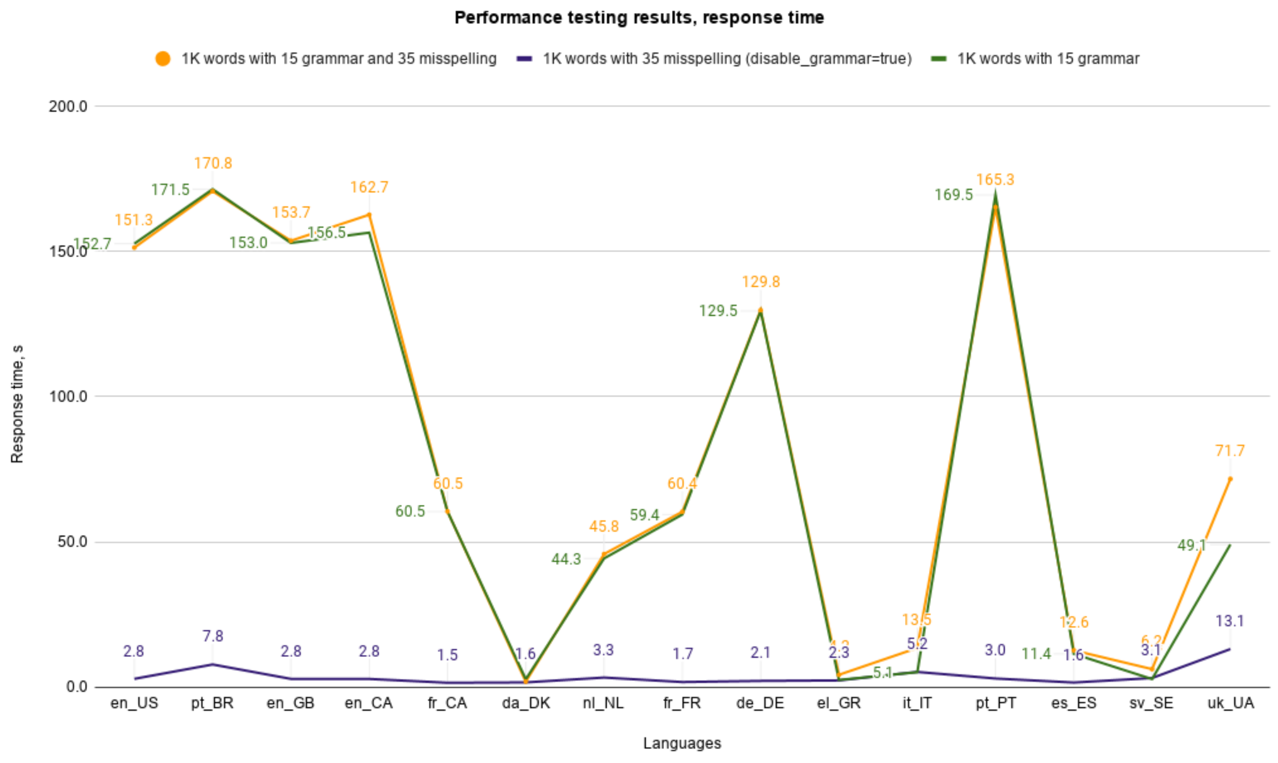


The chart below represents **CPU utilization** results aggregated by language when there are **35 misspellings** and **15 grammar problems** in text of 1K word size.



Summary response time and CPU utilization

The chart below represents **response time** results aggregated by language for 100 simultaneous users in case of three scenarious: 35 mispellings only, 15 grammar problems only, 35 misspellings and 15 grammar problems.



Findings and recommendations

Here are the outcomes and aftermath as well as our advice on hardware and software requirements and notes on performance issues which users may encounter:

> ⚠ During testing we used API requests which contain 1K words per request where 3.5% of words are spelling errors and 1.5% are for grammar errors. On average a person writing on the second language makes around 4%-5% of errors in total. The mechanism of requests distribution and size of each requests for UI-based product are being optimized greatly to ensure smooth experience for users and decrease the load on the servers.

- The response time and CPU untilization metrics depend a lot on the number of words in the dictionary for spelling check and number of rules for grammar check. The more words and rules are for the language, the higher response time and CPU. For example, Brazillian Portuguese contains over 10 million words in the dictionary, thus, it takes longer to process a request in comparison with other languages. The only exception here is the Ukrainian language, it uses a different spelling check engine which doesn't support multi-threading. As a result it has the worst results.
- The spelling check function in more light weight in comparison with the grammar checking. Having grammar checking enabled more than doubles the response time and increases the CPU utilization.
- Under given conditions (size of a text and percentage of errors), when the number of users increases to up 100, one m5 instance entails 99-100% CPU load and a significant increase of response time. Our recommendation for the case when more users are added and CPU load constantly reaches 100% on the machine:
    - upgrade instance type and add more CPUs to it;
    - add one more machine to distribute the traffic (requests) between two or more machines, for example, using load balance and auto-scalling.

If you have any questions or comments regarding the outlined results, please fill free to contact us.