1. main.go:60 Замена regexpa на strings.Contains:
Применяется regex, для поиска просто слова Android (ещё и компилируется на месте)

```
[User359@acer search]$ go test -bench . -benchmem -cpuprofile=cpu.out -memprofile=mem.out -memprofilerate=1
PASS
BenchmarkSlow-8        1        1595516694 ns/op        337004632 B/op    286067 allocs/op
BenchmarkFast-8        2         930143185 ns/op        169674712 B/op    169026 allocs/op
ok      _/home/User359/Documents/mail/go/golang-2017-2/3/99_homework/search     7.064s
```

2. main.go:82 Замена regexpa на strings.Contains:
То же, что в предыдущем пункте, но для MSIE:

```
[User359@acer search]$ go test -bench . -benchmem -cpuprofile=cpu.out -memprofile=mem.out -mempr
ofilerate=1
PASS
BenchmarkSlow-8        1        1482139308 ns/op        337004840 B/op    286122 allocs/op
BenchmarkFast-8        3         356018390 ns/op         3341968 B/op      60530 allocs/op
ok      _/home/User359/Documents/mail/go/golang-2017-2/3/99_homework/search     4.921s
```

3. main.go:25:102 Осуществление замены @ на [at] с помощью strings.Replace, полный отказ от Regexp’ов (скрин производительности потерялся)

4. main.go:19 Чтение всего файла заменил на bufio.Scanner

```
[User359@acer search]$ go test -bench BenchmarkFast -benchmem -cpuprofile=cpu.out -memprofile=mem.out -memprofilerate=1
PASS
BenchmarkFast-8       10         179617792 ns/op         2823980 B/op      38625 allocs/op
ok      _/home/User359/Documents/mail/go/golang-2017-2/3/99_homework/search     3.799s
```

5. seenBrowsers выставляют 2 одинаковых цикла - небольшой рефакторинг с целью уменьшения дублирования кода. На производительность не повлияло.
6. Парсинг json в готовую струтуру User

```
[User359@acer search]$ go test -bench BenchmarkFast -benchmem -cpuprofile=cpu.out -memprofile=mem.out -memprofilerate=1
PASS
BenchmarkFast-8       10         168539229 ns/op         2759836 B/op      37626 allocs/op
ok      /home/User359/Documents/mail/go/golang-2017-2/3/99_homework/search     3.707s
```

7. main.go:82 foundUsers вместо конкатенации строк – пишем в слайс, который при выводе джоиним

```
[User359@acer search]$ go test -bench BenchmarkFast -benchmem -cpuprofile=cpu.out -memprofile=me
m.out -memprofilerate=1
PASS
BenchmarkFast-8       10         162966206 ns/op         2579809 B/op      37550 allocs/op
ok      _/home/User359/Documents/mail/go/golang-2017-2/3/99_homework/search     3.739s
```

8. Парсинг json’а с помощью библиотеки easyjson

```
[User359@acer search]$ go test -bench . -benchmem -cpuprofile=cpu.out -memprofile=mem.out -memprofilerate=1
PASS
BenchmarkSlow-8        1        1487085872 ns/op        337014712 B/op    286499 allocs/op
BenchmarkFast-8       30          68251990 ns/op         2083582 B/op      11553 allocs/op
ok      golang-2017-2/3/99 homework/search      5.364s
```

9. Замена bufio.Scanner на bufio.Reader, чтение сразу []byte без преобразования в строку

```
[User359@acer search]$ go test -bench . -benchmem -cpuprofile=cpu.out -memprofile=mem.out -memprofilerate=1
PASS
BenchmarkSlow-8        1        1489496465 ns/op        337015352 B/op    286574 allocs/op
BenchmarkFast-8       30          58096381 ns/op         1952756 B/op      11542 allocs/op
ok      golang-2017-2/3/99_homework/search      5.086s
```

10. Объединение чтения файла/парсинга json’ов и обработки данных в один цикл (удаление слайса users)

```
[User359@acer search]$ go test -bench BenchmarkFast -benchmem -cpuprofile=cpu.out -memprofile=mem.out -memprofilerate=1
PASS
BenchmarkFast-8       20          51534861 ns/op         1359167 B/op      10528 allocs/op
ok      golang-2017-2/3/99_homework/search      2.966s
[User359@acer search]$
```