

Forum:[Hack Forums](#)

Thread: [Best way to compare two lists of text. Any programming language](#)

SIM-2362662 - February 28, 2021, 6:09 am

I have two very big lists of usernames, list A and B. Each list has 10M usernames

I need a way to remove from list A any username appearing in list B. What would be the fastest way to do this? I'm ok with using any language.

So far, the fastest I managed to do this was using python + numpy. But it was still a matter of tens of minutes. Is there any way to get this done (hopefully a lot) faster?

Dialog - February 28, 2021, 7:24 am

If you are on a POSIX compliant shell, try:

Code `grep -Fvxf B A`

The concept you are referring to is called the "complement" in set theory.

Shadow_Cloud - February 28, 2021, 9:23 pm

(Feb 28, 2021 07:09 AM)? SoFt ? Wrote: [\[CITE\]](#) I have two very big lists of usernames, list A and B. Each list has 10M usernames

I need a way to remove from list A any username appearing in list B. What would be the fastest way to do this? I'm ok with using any language.

So far, the fastest I managed to do this was using python + numpy. But it was still a matter of tens of minutes. Is there any way to get this done (hopefully a lot) faster?

You can probably split these by starting index and compare the indexes in parallel?

Typically the idea is to group by something that will already limit the possible collisions. I think length and starting character should be pretty ideal here, since no two usernames would be the same if

- A) They had a different string length
- B) They don't start with the same character

Since you don't need to apply the operation for each item based on the full list, but only on the string itself you get to create quite a few different groups and since you won't be able to perform this operation as $O(1)$ more sets are better here than trying to increase the speed of a linear operation.

Out of interest, what's the datasource format you have? Is this is plain text files? If so, how big are the source files?

SIM-2362662 - March 1, 2021, 4:20 am

(Feb 28, 2021 10:23 PM)Shadow_Cloud Wrote: [\[CITE\]](#) You can probably split these by starting index and compare the indexes in parallel?

Typically the idea is to group by something that will already limit the possible collisions. I think length and starting character should be pretty ideal here, since no two usernames would be the same if

- A) They had a different string length
- B) They don't start with the same character

Since you don't need to apply the operation for each item based on the full list, but only on the string itself you get to create quite a few different groups and since you won't be able to perform this operation as $O(1)$ more sets are better here than trying to increase the speed of a linear operation.

Out of interest, what's the datasource format you have? Is this is plain text files? If so, how big are the source files?

Yeah, the files are plain .txt

A 10M list has around 700Mb in size.

This sounds like a good idea, but I'm not sure how to apply something like that

My plan after reading your post is now to sort one of the lists alphabetically and then use use divide et impera to check for each username in the second list if its bad or not. What do you think? Do you see any way to improve this?

Shadow_Cloud - March 1, 2021, 7:42 am

(Mar 1, 2021 05:20 AM)? SoFt ? Wrote: [\[CITE\]](#) Yeah, the files are plain .txt

A 10M list has around 700Mb in size.

This sounds like a good idea, but I'm not sure how to apply something like that

My plan after reading your post is now to sort one of the lists alphabetically and then use use divide et impera to check for each username in the second list if its bad or not. What do you think? Do you see any way to improve this?

So lets use a pretend subset of the lines to illustrate what I mean before we get into the code

Code List A

```
admin
root
administrator
sysadmin
wildfly
jboss
appserver
serveradmin
servadmin
deployadm
```

List B

```
root
admin
test
guest
info
adm
mysql
user
administrator
oracle
```

Instead of comparing the first value in List A with every single value in List B, you can Group each set by the starting character, so you only need to compare the words in List B that starts with the same first letter, because if they don't, you already know its not a duplicate, the same goes for the length of the string.

Its probably easier to process this as stuffing this all into a new object called AllData and doing a

Group By on that, something like this:

```
Code var myList = AllData.GroupBy(g=> new {g.userName})  
    .Where(w=>w.Count() > 1)
```

If you really want to get into parallel processing you'll have a different ballpark to get into. What's more optimal is also going to depend on your hardware and how much compute power you have available. Generally this would be approach I suggest though.

SIM-2362662 - March 1, 2021, 4:32 pm

(Mar 1, 2021 08:42 AM)Shadow_Cloud Wrote: [\[CITE\]](#) So lets use a pretend subset of the lines to illustrate what I mean before we get into the code

Code List A

```
admin  
root  
administrator  
sysadmin  
wildfly  
jboss  
appserver  
serveradmin  
servadmin  
deployadm
```

List B

```
root  
admin  
test  
guest  
info  
adm  
mysql  
user  
administrator  
oracle
```

Instead of comparing the first value in List A with every single value in List B, you can Group each set by the starting character, so you only need to compare the words in List B that starts with the same first letter, because if they don't, you already know its not a duplicate, the same goes for the length of the string.

Its probably easier to process this as stuffing this all into a new object called AllData and doing a Group By on that, something like this:

```
Code var myList = AllData.GroupBy(g=> new {g.userName})  
    .Where(w=>w.Count() > 1)
```

If you really want to get into parallel processing you'll have a different ballpark to get into. What's more optimal is also going to depend on your hardware and how much compute power you have available. Generally this would be approach I suggest though.

Ah I see. Wouldnt it be as quick (if not slower) as this sort method I just explained? I'm guessing its about the same idea at the core : classifying the data so you can use different tricks when searching for a string.

Anyway, I coded this method in python (sort list + divide et impera) and its perfect! In just 19

seconds I can get the common elements from 2 lists of 4M users. I'm super pleased with the result. Thank you so much for the input!

Shadow_Cloud - March 1, 2021, 8:47 pm

(Mar 1, 2021 05:32 PM)? SoFt ? Wrote: [\[CITE\]](#) Ah I see. Wouldnt it be as quick (if not slower) as this sort method I just explained? I'm guessing its about the same idea at the core : classifying the data so you can use different tricks when searching for a string.

Anyway, I coded this method in python (sort list + divide et impera) and its perfect! In just 19 seconds I can get the common elements from 2 lists of 4M users. I'm super pleased with the result. Thank you so much for the input!

Glad it worked :)

You'll find that different approaches are going to work better at different "tipping points" depending on the size of the data.

I spend a lot of time optimizing code and this is a pretty common way to speed up processing, distribute the workload across multiple nodes. Using things like docker and k8s you can easily spin up and spin down nodes (or pods in this case)

There's a difference depending on whether you have a once off requirement or whether you'll expand the data set in future and want to run this against a growing data set, since expanding on the set in future and testing for a single value will be much faster the more you've narrowed down the set. Effectively this is an $O(n)$ comparison and rather than speeding up the algorithm being used to $O(1)$ or $O(\log(n))$ you are just minimizing the dataset being pushed as n .

sangita123 - March 2, 2021, 6:53 am

(Mar 1, 2021 05:20 AM)? SoFt ? Wrote: [\[CITE\]](#) Yeah, the files are plain .txt

A 10M list has around 700Mb in size.

This sounds like a good idea, but I'm not sure how to apply something like that

My plan after reading your post is now to sort one of the lists alphabetically and then use divide et impera to check for each username in the second list if its bad or not. What do you think? Do you see any way to improve this?