**Java example: Solution starts on page 2**

**Total 8 pages explanation & details of references**

# Objective of StringArray ( Optimization of code Performance)

small sample of your code and to see how you approach and solve a simple problem. Ideally

This is a simple class to extract matching strings from an array regardless of the characters

order.

You will write a Java class that accepts an array of strings in the constructor. The class will

expose a find function that accepts a string. The function will return all strings from the array

that contains the exact same characters as the string passed into it.

The order of the characters in the strings is not relevant.

For example, the constructor should take an array as follows:

String[] thisIsAStringArray = new String[5];

thisIsAStringArray[0] = "asd";

thisIsAStringArray[1] = "asdd";

thisIsAStringArray[2] = "fre";

thisIsAStringArray[3] = "glk";

thisIsAStringArray[4] = "lkm";

Finder finder = new Finder(thisIsAStringArray);

Calling finder.find('sad') should return an array containing the string asd .

In the case where more than one member of the initialization array matches the key the return

array will have more than one member.

Requirements

State your assumptions. Anywhere you feel that the requirements are unclear please

make an assumption and document that assumption.

Please comment your code where needed. Find the right balance between comments and

self-documenting code. Code quality is of high importance.

You are not allowed to use any external libraries aside from jUnit. Gradle or maven are

permitted for integrating the latter.

The performance of your code matters, so please try to keep your code as optimized as

possible.

Write tests to showcase the quality of your solution, its performance (e.g. 50k+ runs, large

initialization arrays etc.) and its edge cases.

Java 8

Solutions to the problem:

Deliverables

**Question: Java source-code and tests**

**Solution:** Attached in the Note pad: the email: FinaljavacodeArrayStringFinder

Instructions on how to compile your project or a script/build step that does it

Automatically --- Instructions : <https://github.com/twitter/Serial/commit/200febb6bc3878bdbee7f4d16cea5199eb58edbd>

Any other material that you feel is relevant for your solution

Few example that could to considered for expansion of the logic that can be considers if the use case changes

<https://www.javaguides.net/2018/08/how-size-of-arraylist-increases-dynamically.html>

<https://dzone.com/articles/java-convert-array-to-string?fromrel=true>

# Big Data Deep-Dive

**Question: Understand your perspective on big data systems. To achieve this, you are tasked with creating a high-level diagram of a big data system that solves the challenge of applying sentiment** analysis to a real-time stream of tweets.

**Solution:** Architecture in the diagram below

**Question: The sentiment analysis model is pre-trained and available**

Solution: Available with in twitter: As for analysis

Sentiment analysis or opinion mining tries to find emotion hidden behind the text. Complex algorithms aim to determine whether the author of a tweet was sad, happy, angry or something else. It is of special interest to determine whether the author writes positive or negative about whatever the subject of the tweet is, as this can be used for marketing purposes, managing reputations or analyzing popularity.

**Question: sentiment analysis models are available for all languages:**

<https://developer.twitter.com/en/docs/twitter-for-websites/twitter-for-websites-supported-languages/overview>

**Question: you have direct access to the twitter firehose, outputting ~6k tweets per second there is an SDK available that allows you to connect to the firehose**

Solution: Support the above Requirement:

<https://developer.twitter.com/en/docs/tweets/sample-realtime/api-reference/decahose#Methods>

Requirements

**the system is real-time – Real time Analysis in diagram**

**the system has a way to cope with back pressure**

**Solution:** Enter Reactive Streams. It allows the developers to have a well throttled (back-pressured) flow of data throughout such systems. It gained popularity, and since it became a standard [various independent libraries](https://github.com/search?l=Java&q=reactive+streams&type=Repositories&utf8=%E2%9C%93) speak the same semantics, and can seamlessly connect to each other.

the system routes tweets to the correct model depending on language

the system performs basic feature engineering as follows:

**Question: converting emoticons / smileys to tokens:**

**Solutions:** Emoticons as language independent indicators for sentiment Of all the text messages sent on Twitter each day, approximately 5 - 10% contains an emoticon. These are used to express a wide range of different emotions and determine in most cases correctly the sentiment the author conveyed with his message, making it ideal to use as noisy labels for distant supervised learning. Of particular interest are emoticons that express positive or negative emotions, such as sadness, anger, happiness or delight. Some examples of typical western emoticons can be found in below. It is to be kept in mind, that eastern languages more often use non-rotated symbols, which would also have to be included if these languages are of interest, but can mostly be ignored otherwise. Negative :( :-( :{ :-|| :@ :'( :/ D:< ಠ\_ಠ Positive :-) :) :o) :-D :D 8-D XD =) Table 1 Typical emoticons to express positive and negative sentiment in western languages The Twitter streaming API allows any developer to access part of the huge amount of data on Twitter and the search can additionally be specified as to only contain tweets with certain keywords or emoticons and as of May 2013 even the language can be specified. If enough time is available, building a database with a huge amount of tweets can be done very easily and the requirements to the processor are such that even a simple singleboard computer such as a Raspberry Pi is capable of fetching millions of tweets every day. While positive emoticons are more common, it is still advisable to generate a corpus of approximately the same size for positive and negative texts to simplify further analysis. At this stage, it is also important to have the language information of every text available or separate every language that is to be analyzed into its can be language-specific lists of insults or swear words to further indicate a negative sentiment. When combining this information with the sentiment lexicon, all words which appear in the gathered lists as positive should have their value for positive occurrences increased, while for negative words the value for negative occurrences should increase. The exact values should correlate to the size of the sentiment lexicon in the given language. Given an equal amount of positive and negative training data, there is an additional issue to be kept in mind, namely the way people use the emoticons. In most languages, positive emoticons are about four times more often than negative ones. People are frequently using positive emoticons to convey that they are happy even if the text itself would not indicate it in any way. This means that a classification of a word into the negative class is stronger than one into the positive class where it is more “washed-out”. Either the algorithm used for classification of a text will reflect that later on, or some adjustments have to be made directly in the sentiment lexicon. The strength of this effect can be measured by using a test set with an identical amount of negative and positive tweets in the given language. Ideally, a classifier based on the given sentiment lexicon should have approximately the same amount of false positives for negative tweets, as well for positive tweets in this situation. As an example, in case of the naïve Bayes algorithm [17] with an English test data set, about two thirds of the tweets were classified as negative and one third as positive. The most direct and simplest way to adjust for this problem proved to be multiplying the amount for “total amount of positive texts” by a value of 1.3, resulting in an approximately even distribution between tweets classified as positive and negative. This value varies between languages, so if possible, the test should be repeated with every used language. There is one further adjustment to the sentiment lexicon that might be useful. Since rarely used words are not getting deleted, they have a high chance to become victims of random effects with a strong impact on the classification. For example any uncommon word which was supposed to be neutral, but through random effects had a high amount of co-occurrences with positive smileys is not a reliable indicator of sentiment. To counter this effect, a small number, proportional to the total amount of tweets in the given language should be added to all negative and positive occurrences.

**Question: converting GPS coordinates if available to City + Country:**

**Solution :** <https://developer.twitter.com/en/docs/geo/place-information/api-reference>

**Question: The system saves data to the database system of your choice the system and the storage backend can scale horizontally**.

**Solution:** Backend storage systems that could be used are Amazon S3, Cassandra, Mongo DB Snowflake & Amazon Redshift

Deliverables

**Question: High-level description of the system (2-3 sentences)**

**Solution:** Abstract Sentiment analysis is an important tool in the study of social media and is very well researched for texts written in English. However, in many cases multi-language text analysis is required and a simple translation of the text to English would result in inferior solutions. A novel field of application is the analysis of the communication in social media by politicians in a country with multiple national languages, such as Switzerland.

**Question: high-level system diagram (can be hand-drawn and photographed, no aesthetic requirements beyond readability**) Attached diagram

description of libraries / frameworks chosen to solve the task (bullet points + 1 sentence

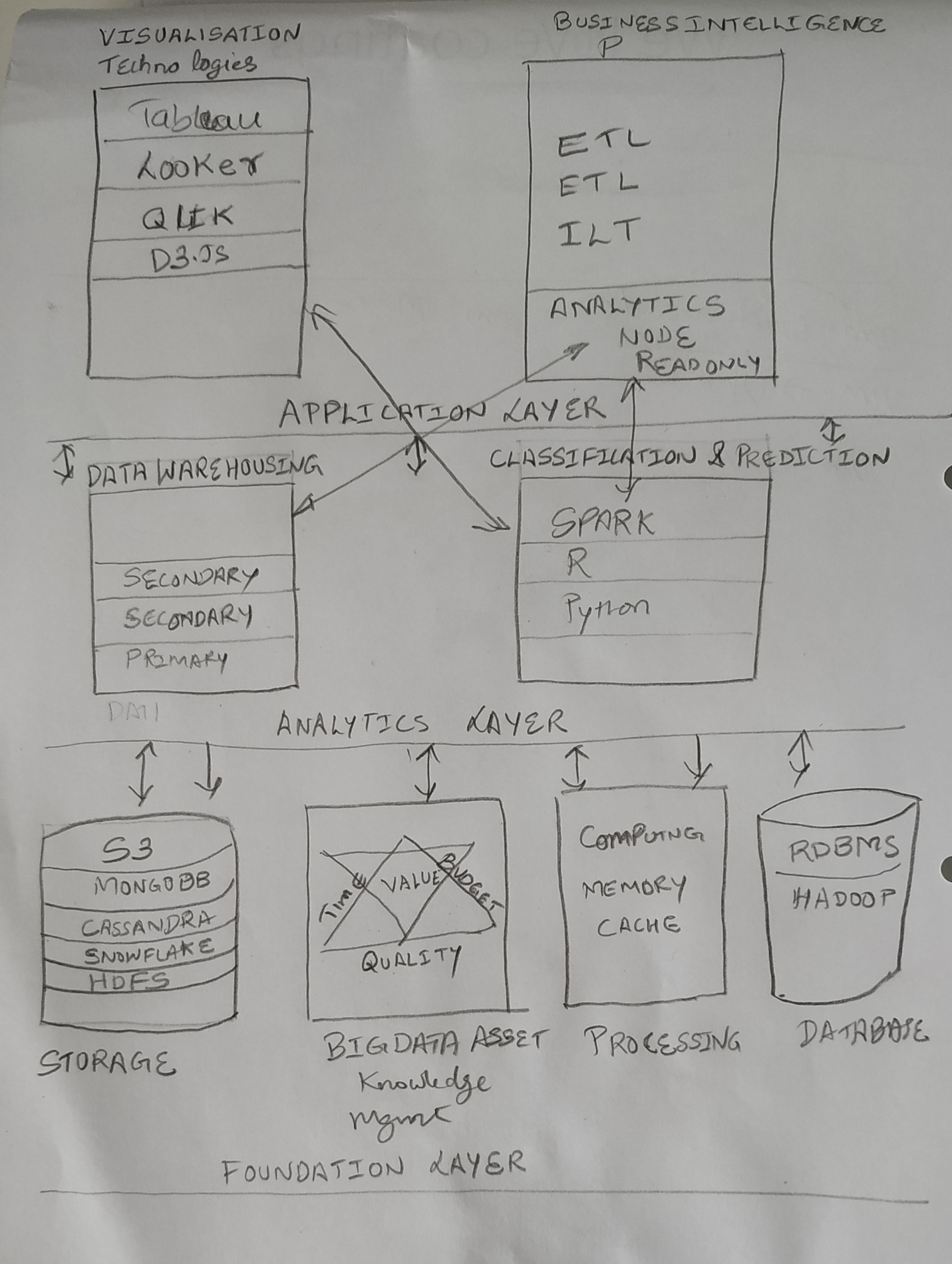
for each suffice) <https://developer.twitter.com/en/docs/developer-utilities/twitter-libraries>

any observations that you feel are relevant Notes

**Question: coding is not required**

**logic such as routing can be expressed using boxes and arrows technology choices can be based on what's available or what you're familiar** with there is generally no single right way to solve a task: focus on explaining your choices rather than finding the perfect tool.

Solution: details in the diagram below. Explain in 3 different Levels





Detailed Architected diagram form Sentiment Analysis: Clear Diagram is also Attached in the email.

