# Ashish Sharma's Tech Blog

This blog contains series of posts explaing basic and advanced concepts in data structures, algorithms, parallel processing, system design along with fundamentals in Java and unix with the sample code.



Tuesday, January 31, 2012

## Spelling Corrector Algorithm - Java Code

I was amazed at how Google does spelling correction so well. Type in a search like 'speling' and Google comes back in 0.1 seconds or so with Did you mean: ***spelling***. Impressive!

In one of my past projects, I was working on a RealEstate infrastructure where we had to support spelling corrections for City/State searches. At that point, we used Solr's spelling corrector. But I wanted to dig into the algorithm and write something simple and works without Solr. Just did some research online, went through some papers publishings and wrote an algorithm that serves what we are getting from Solr.

The code maintains a static dictionary of correct spellings in Hashmap, which we load on-boot. Below is the algorithm written in Java.

It works in O(n), n being the length of the word.

**Spelling Corrector - Java code**

```
1  package main.java.algo;
2
3  import java.util.ArrayList;
4  import java.util.Collections;
5  import java.util.HashMap;
6  import java.util.List;
7  import java.util.Map;
8
9  // References
10 //   - http://norvig.com/spell-correct.html
11 //   - http://raelcunha.com/spell-correct.php
12 //   - http://surguy.net/articles/scala-spelling.xml
13
14
15 // Components
16
17 //- dictionary (pre populated), map with word to weight
18 //- edits : deletes, transpose, replace, inserts
19 //- candidates : loop edits and populate. map to weight to word
20
21 public class SpellingCorrector {
22
```

## Search

Search

## Programmer | Blogger | Traveller

**Ashish Sharma**

Cupertino, CA, United States

Hi! I am a software developer; working with Apple Inc.; specializing in Java, J2EE technologies; data structures & algorithms, architecture & design, with big data on the backend infrastructure; expertise in Hadoop, Cassandra, Solr, Oracle & MySQL.

Over 8 years of experience in developing highly scalable algorithms & applications running over terabytes of data serving millions of read/writes a second. I have worked across entire stack of technologies using Java, Servlets, Filters, JSP, Tuckey, Multi-threading, Caching, Regex, XML, JSON, MySQL, Solr, Lucene, jQuery, JavaScript on apache servers httpd, tomcat, with big data on the backend living on HDFS and CDFS.

Through this blog, I'll try to share my learnings. Hope it helps!

View my complete profile

## Source Code

```
23  public static void main(String[] args) {
24
25   dictionary.put("spelling", 1);
26   dictionary.put("ashish", 1);
27   dictionary.put("param", 1);
28
29   String input = "spiling";
30   String correct = correct(input);
31   System.out.println(correct);
32  }
33
34
35  // word to count map - how may times a word is present - or a weight attached to a word
36  public static Map<String, Integer> dictionary = new HashMap<String, Integer>();
37
38  public static String correct(String word) {
39
40   if(dictionary.containsKey(word))
41    return word;
42
43   // getting all possible edits of input word
44   List<String> edits = edits(word);
45
46   // Here we can either iterate through list of edits and find the 1st word that is in dictionary and return.
47   // Or we can go to below logic to return the word with most weight (that we would have already stored in dictionary map)
48
49   // map to hold the possible matches
50   Map<Integer, String> candidates = new HashMap<Integer, String>();
51
52   // keep checking the dictionary and populate the possible matches
53   // it stores the count (or weight) of word and the actual word
54   for(String s : edits) {
55    if(dictionary.containsKey(s)) {
56     candidates.put(dictionary.get(s), s);
57    }
58   }
59
60   // one we have found possible matches - we want to return most popular (most weight) word
61   if(candidates.size() > 0)
62    return candidates.get(Collections.max(candidates.keySet()));
63
64
65   // If none matched.
66   // We will pick every word from edits and again do edits (using same previous logic) on that to see if anything matches
67   // We don't do recursion here because we don't the loop to continue infinitely if none matches
68   // If even after doing edits of edits, we don't find the correct word - then return.
69
70   for(String s : edits) {
71
72    List<String> newEdits = edits(s);
73    for(String ns : newEdits) {
74     if(dictionary.containsKey(ns)) {
75      candidates.put(dictionary.get(ns), ns);
76     }
77    }
78   }
79   if(candidates.size() > 0)
80    return candidates.get(Collections.max(candidates.keySet()));
81   else
82    return word;
83  }
84
85  // Word EDITS
```

The source code of all examples and implementations in posts can be found at My GitHub Repository. I'll keep updating as I write more or fix issues.

## Labels

Abstraction algorithms big-data binary tree binary-operations bit-operations BreadthFirstSearch bubble-sort cache comparable comparator complete tree concepts data-structures database Encapsulation external-sort garbage-collection graphs hadoop heap hexadecimal inner-class insertion sort internationalization inverted-index java jdbc joins linked-list LinkedHashMap LRU lucene mathematical computation memory merge-sort mutex mysql oops overload override polymorphism problems queue quick-sort recursion reflection selection sort semaphore serialization shift-operators sorting stacks static stringbuffer stringbuilder strings system-design threads tree tree traversal unix

## Popular Posts

### Understanding Java Heap Space and Memory Tuning

What lies on Heap - Instance variables (including instance level references to objects) and the Objects lie on HEAP. This includes the Ob...

### External Merge Sort

External sorting is required when the data being sorted do not fit into the main memory of a computing device (usually RAM) and instead the...

### Binary Trees Traversal Algorithm - Inorder, Preorder, Postorder

Traversals of Binary Trees There are several well-known ways to traverse a binary tree. Let us go through Inorder Traversal, Preorder Tra...

### Binary Tree Implementation in JAVA

Before we proceed with the Binary tree implementation, let us first understand the building elements - nodes. Lets define the data structure...

### Spelling Corrector Algorithm - Java Code

```
86   // Getting all possible corrections c of a given word w.
87   // It is the edit distance between two words: the number of edits it would take to turn one into the other
88
89   public static List<String> edits(String word) {
90
91    if(word == null || word.isEmpty())
92     return null;
93
94    List<String> list = new ArrayList<String>();
95
96    String w = null;
97
98    // deletes (remove one letter)
99    for (int i = 0; i < word.length(); i++) {
100    w = word.substring(0, i) + word.substring(i + 1); // ith word skipped
101    list.add(w);
102   }
103
104   // transpose (swap adjacent letters)
105   // note here i is less than word.length() - 1
106   for (int i = 0; i < word.length() - 1; i++) { // < w.len -1 :: -1 because we swapped last 2 elements in go.
107    w = word.substring(0, i) + word.charAt(i + 1) + word.charAt(i) + word.substring(i + 2); // swapping ith and i+1'th char
108    list.add(w);
109   }
110
111   // replace (change one letter to another)
112   for (int i = 0; i < word.length(); i++) {
113    for (char c = 'a'; c <= 'z'; c++) {
114     w = word.substring(0, i) + c + word.substring(i + 1); // replacing ith char with all possible alphabets
115     list.add(w);
116    }
117   }
118
119   // insert (add a letter)
120   // note here i is less than and EQUAL to
121   for (int i = 0; i <= word.length(); i++) { // <= because we want to insert the new char at the end as well
122    for (char c = 'a'; c <= 'z'; c++) {
123     w = word.substring(0, i) + c + word.substring(i); // inserting new char at ith position
124     list.add(w);
125    }
126   }
127
128   return list;
129  }
130
131 }
```

Posted by Ashish Sharma at 9:31 AM



Labels: algorithms, java

## 22 comments:

**Suha** October 9, 2012 at 8:57 AM

Thank you very much very helpful indeed

Reply

I was amazed at how Google does spelling correction so well. Type in a search like 'speling' and Google comes back in 0.1 seconds o...

Binary Heap: Concepts & Implementation in Java

package main.java.algo.sorting.heap; /****** This class is about Priority Queue, which is implemented using Binary Min Heap. *** So h...

Construct a Tree from Inorder & Preorder traversals

This post is in continuation to my earlier post on Build a Binary Tree, given two traversals The following procedure demonstrates on how ...

Semaphores and Mutexes

The Classic Toilet Example: Mutex : Is a key to a toilet. One person can have the key - occupy the toilet - at the time. When finished, ...

Inorder Traversal Without Recursion using Stacks

Logic for Inorder traversal without recursion, using a stack. Pseudo code :: inOrderTraverse (Node root) { -> Stack s -...

Creating a Custom LRU Cache in Java

Caching is an extremely useful concept to improve performance. There are many commercial and open-source caching packages available for Java...

## Blog Archive



## Followers

**Abhiroop Sarkar** December 27, 2013 at 5:13 AM

// one we have found possible matches - we want to return most popular (most weight) word
if(candidates.size() > 0)
return candidates.get(Collections.max(candidates.keySet()));

I think this step would serve better if you use the Bayes theorem to find which correction is more likely to occur instead of just assuming the most frequent word in the dictionary to be the correction.

Reply

**bsjug** July 2, 2015 at 9:40 AM

Dictionary populated can be in anothe class.

Reply

**Allsearchengines.blogspot.com** September 22, 2015 at 10:23 AM

This is the best lead I've come across on how to build a grammar checker.

Reply

**Dhiya L** March 23, 2016 at 1:50 AM

Thank you for sharing such a nice story with us. I really enjoyed very much And please keep updating like this with this site.

Java Training

Reply

**Priya Tamil** March 23, 2016 at 2:27 AM

Great work, programming was very easily understand and more important coding are provided on this post and this is very valuable in my studies,all coding easily understand and develop more skills,thanks for sharing this post.
php training

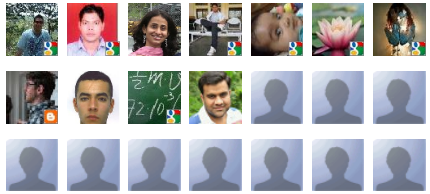
Reply

**Priya Tamil** March 25, 2016 at 3:39 AM

Another great articles and very interesting to read,thanks for sharing that wonderful useful information,given programming coding was very excellent and i can easily observe all provided information.
sas

Reply

**Seguidores (22)** Próxima



Seguir

**deeksha** March 27, 2016 at 10:38 PM

your blog is really helpful and useful it really helped me during coding process and it is useful thanks for sharing those information.

digital marketing training in chennai

Reply

**Haritha** April 2, 2016 at 1:44 AM

Thank you for your post. This was really an appreciating one. You done a good job. Keep on blogging like this unique information with us.

SAS Training in Chennai

Reply

**Unknown** April 10, 2016 at 1:21 PM

This only considers words at an edit distance of 1 from the original word, is that correct?

Reply

**Rekhila Pk** April 12, 2016 at 1:50 AM

Great blog..Step by step explanation is too good to understand..Its very useful for me to understand..Keep on sharing..
SEO Training in Chennai

Reply

**Priscilla** April 12, 2016 at 7:04 AM

Wow amazing i saw the article with execution models you had posted. It was such informative. Really its a wonderful article. Thank you for sharing and please keep update like this type of article because i want to learn more relevant to this topic.

SAP Training in Chennai

Reply

**Priscilla** April 14, 2016 at 10:17 PM

Thank you for sharing the explanation which is related with spelling corrector. It is much essential for typing through the document. So please keep update like this.

SAP Training in Chennai

Reply

**Harini R** April 15, 2016 at 5:30 AM

thanks for sharing wonderful information keep sharing more.
Software Testing Training in Chennai

Reply

**Nainika Joseph** May 4, 2016 at 4:08 AM

Really very nice blog information for this one and more technical skills are improve,i like that kind of post.

SAP training in Chennai

Reply

**Camellia Canan** June 1, 2016 at 8:22 AM

Thanks for the good words! Really appreciated. Great post. I ve been commenting a lot on a few blogs recently, but I had nt thought about my approach until you brought it up.

SAP training in Chennai

Reply

**Nainika Joseph** June 2, 2016 at 12:19 AM

All are saying the same thing repeatedly, but in your blog I had a chance to get some useful and unique information, I love your writing style very much, I would like to suggest your blog in my dude circle, so keep on updates.

SAP training in Chennai

Reply

**Sathya G** June 15, 2016 at 4:43 AM

Thanks you for sharing the unique content. you have done a great job. thank you for sharing such a unique content.
Seo Company in Chennai

Reply

Replies

**Training Coimbatore** June 26, 2016 at 6:44 AM

great post

Hadoop training in coimbatore
Java training in coimbatore
Oracle training in coimbatore
Informatica training in coimbatore
Oracle training in coimbatore
Informatica training in coimbatore

**Reply**

**Suseela Susiee** July 1, 2016 at 6:36 AM

Great and useful article. Creating content regularly is very tough. Your points are motivated me to move on.

SEO Company in Chennai

Reply

**Unknown** July 11, 2016 at 6:24 AM

Excellent Article

Reply

**Kr** September 22, 2016 at 4:01 AM

how to do phrase and sentence spell check
I mean context-sensitive

Reply

Enter your comment...

**Comment as:** - (Google) ▾ **Sign out**

Publish **Preview** Notify me

Home Older Post

Subscribe to: Post Comments (Atom)