# Implementation of basic spell suggester like Google’s “Did you mean?”

**Introduction**:

“Search” has become an implicit part of any web application. There are many packages, libraries, software and extensions out there which help implementing a functional and custom search engine for our web application. However, only a few of those are really intelligent enough to detect a wrong spelling and provide a probable suggestion.

We definitely don’t want our visitors to leave our site just because of their misspelled search queries. Imagine a grocery e-commerce website which would not give a search suggestion of “Onion” when queried “Anion”, “Mayonnaise” when queried “Mayonis”, “ladies’ finger” when queried “lady's finger” etc.

And this leads us to have a look on one of the libraries that is becoming incredibly handy to implement spell suggester in web application and text editors. This library is officially used by various text editors including Libre Office, Open Office, Firefox, Chrome, MacOS, and Opera etc. I am talking about ‘Hunspell’ library. Today, however, we will just focus on how to get a basic spell suggester like Google’s “Did you mean?” response using ‘**NHunspell** library’, which is a Hunspell library for dot net platform.

Basically, in the deepest abstraction, (N)Hunspell needs two files:

1. Dictionary file (.dic)
2. Affix file (.aff)

Dictionary file (.dic) stores the set of valid and properly spelled words of any language which Hunspell uses for reference. Affix file (.aff) contains rules of getting proper and valid suggestion by assigning prefix or suffix to the input (and potentially misspelled) word. So if you want to implement spell suggester for a standard language like English, you will get tons of dictionary (.dic) files and related (.aff) files online.

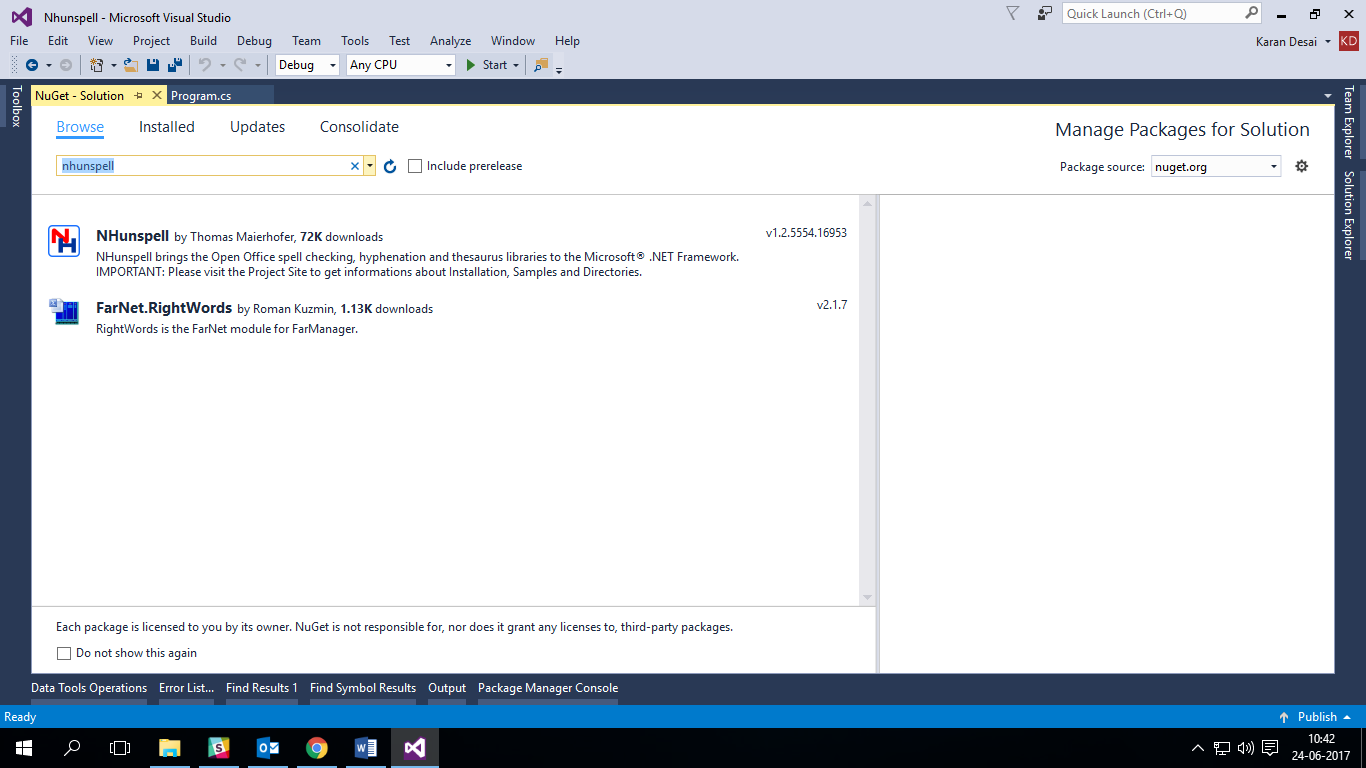
However, if you want Nhunspell to suggest you custom words (say a misspelled product name for a grocery store) it becomes slightly complex because you have to teach Hunspell that *this*  is my ‘*custom’* language and *this* is my dictionary (with valid names and spelling of products), and suggest me correct words using *these* rules (mentioned in custom affix file). And this tutorial is exactly for this purpose. We will learn to use Nhunspell to implement a custom dictionary in a simple console application. The tutorial is very basic, so treat it as just your starting point.

If you are interested in implementing spell suggestions for standard languages instead, you can go [here.](http://www.techrepublic.com/blog/software-engineer/spell-check-net-applications-with-nhunspell/)

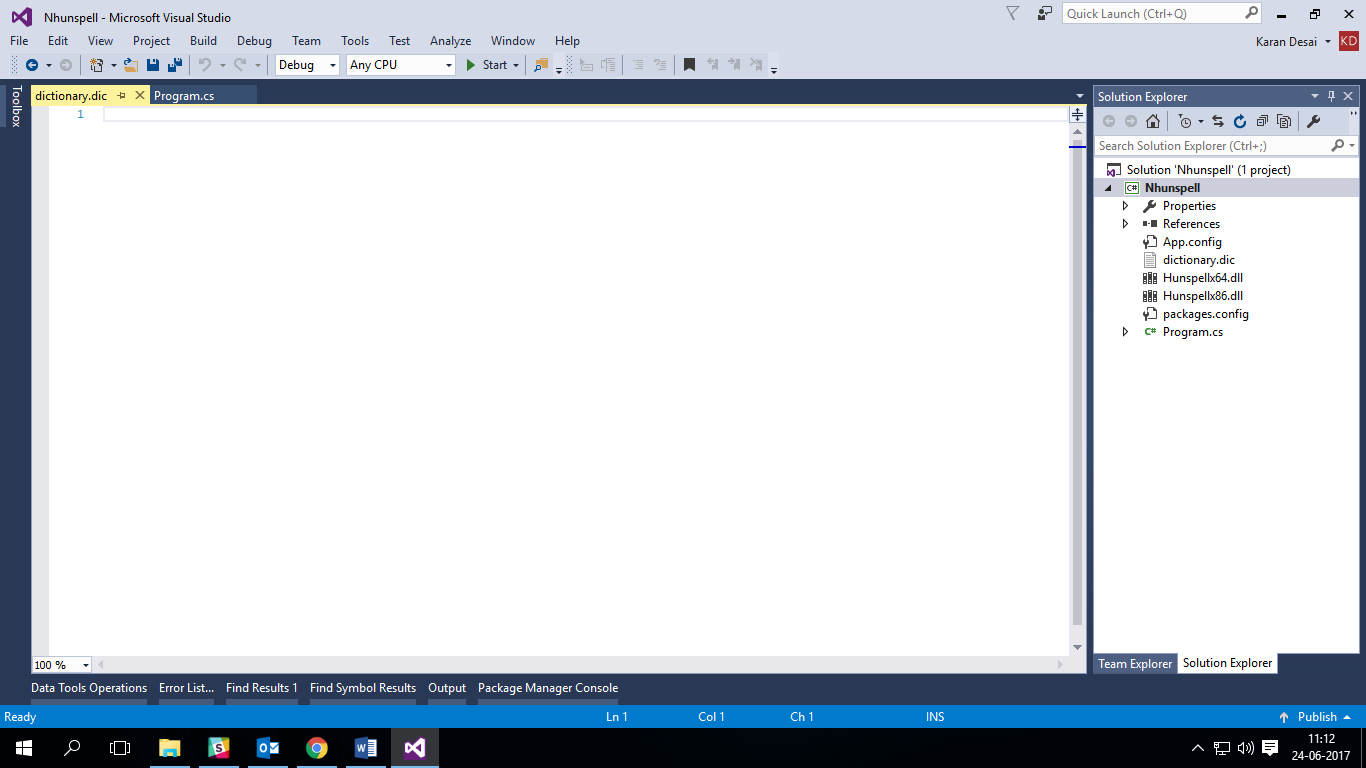
Let’s get started:

**Step 1:** Create a new console project from Visual Studio.

**Step 2:** From Tools>NuGet Package Manager >Manage NuGet Packages for solution, browse the Nhunspell library and install it.

`

**Step 3**> Add a text file by right clicking your project name> Add item with .dic extension. Give it a name say, dictionary.dic



.dic is generally a file with a list of valid names and words with usually a total at the top. For example,



Here, 5 indicates total words in this dictionary and each word from next line is the valid custom word that Hunspell would use during spell checking.

Step 4> Now add another text file in the project with .aff extension. This file will contain set of rules which Hunspell would use for checking and suggesting.

The basic structure of affix file for the suggestion is as follows:

SET encoding->set character encoding of word (Possible values are: UTF-8, ISO8859-1 etc)

TRY characters-> Hunspell can take characters specified in Try to replace with misspelled word to get the correct word. For example, following command

TRY a|e|i|o|u

Will try to place above specified vowels in misspelled words to get the correct word. So if you spelled union or anion, it will suggest onion.

REP -> Replace characters. Specifies rule to replace a character chunk with another. For example, following commands

REP 2

REP ed y

REP y ed

Indicates that if *you* find y try to replace it with ed. So it will suggest Baked Buns for Bakery Buns.

So enough of talking, it's time for action:

Jot down few words in your dictionary.dic file:



Jot down few rules in affix.aff file:



I have added very basic rules (of mistyping due to slip of finger in ‘qwerty’ keyboard in TRY block and replace ‘an’ with ‘on’ for onion)

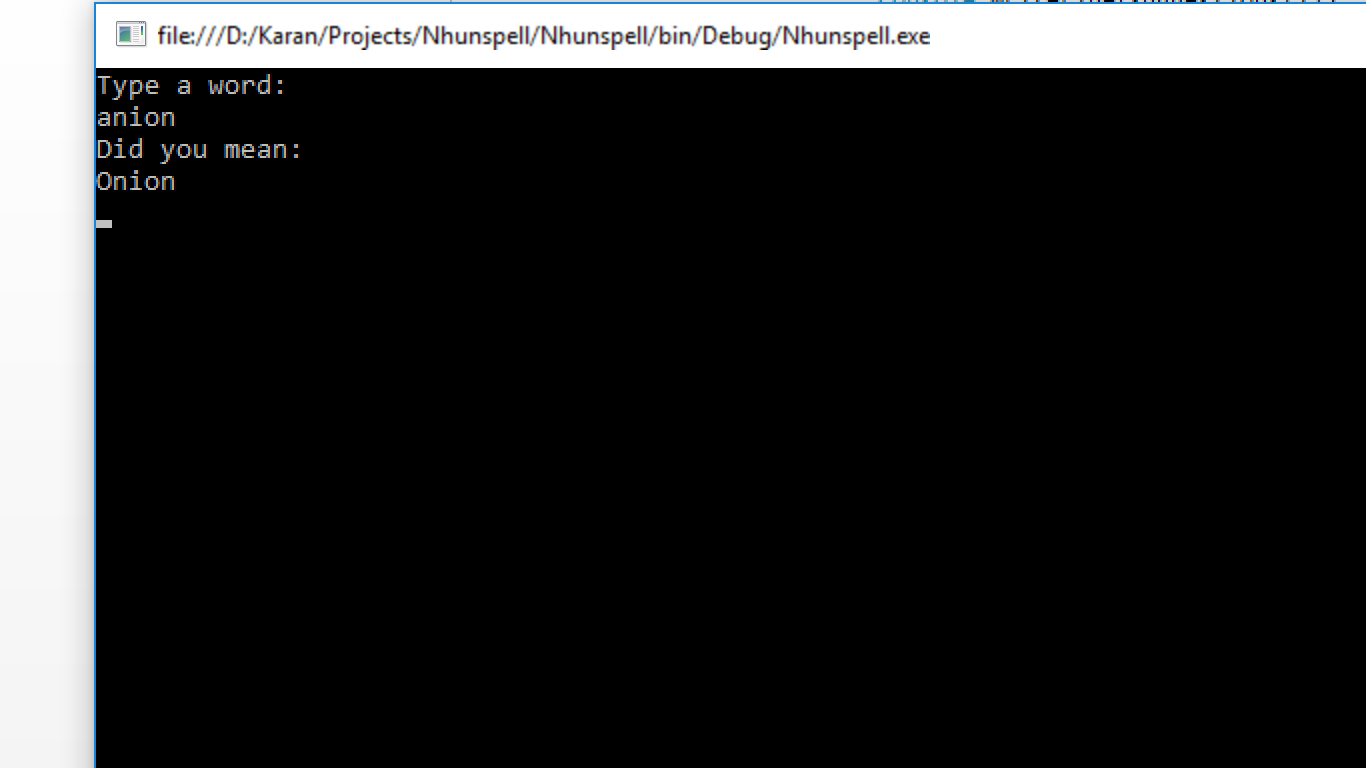
Step 5> Code: Note the following inbuilt function from Nhunspell library. (Add using Nhunspell;):

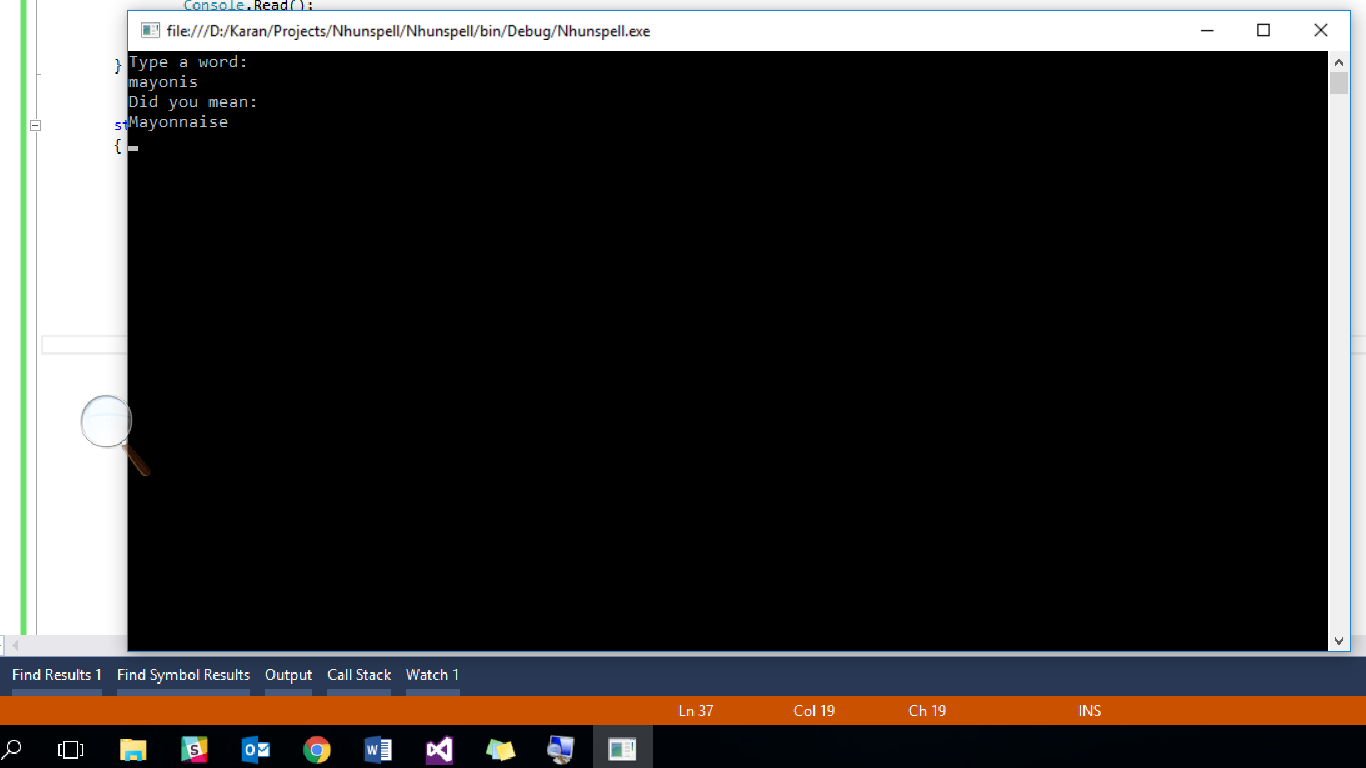
1. The hunspell instance is created from Hunspell class
2. Load method loads affix and dictionary files.
3. Suggest method takes the input word and checks it with loaded dictionary and affix files and returns a list of probable words.

That’s it! It’s that simple. This library is a sugar coating to the rather bitter and tedious process!



And how will the output look, run the project to see the magic:





Congratulations, you just successfully made your custom dictionary spell suggester!

Want to play with source code? Fork my repository: [Nhunspell-Implementation-Console-Application-with-custom-dictionary](https://github.com/karandesai28/NHunspell-Implementation-in-Console-Application-with-custom-dictionary)