Preprocessing_sc1

October 15, 2020

[128]:
```
import numpy as np
import pandas as pd
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
import re
from tqdm.auto import tqdm
import joblib
from bs4 import BeautifulSoup
from nltk.stem.porter import PorterStemmer
import time
from sklearn.metrics.pairwise import cosine_similarity
import smart_open
import gensim
import pickle
import warnings
warnings.filterwarnings('ignore')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[ ]:
```
df = pd.read_csv('/content/drive/My Drive/self_case_study1/data/all_data.csv')
df.head()
```

[ ]:

	id	title	…	score	tag_count
0	9	In C#, how do I calculate someone's age based …	…	4502	3
1	11	Calculate relative time in C#	…	2327	5
2	126	How would you access Object properties from wi…	…	262	4
3	330	Should I use nested classes in this case?	…	63	4
4	742	Class views in Django	…	73	4

[5 rows x 7 columns]

[ ]:
```
df.shape
```

[ ]:
```
(572554, 7)
```

[ ]:
```
# below mentioned functions are helper functions which is use for text␣
 ↪preprocessing
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    phrase = re.sub(r"\n", "", phrase)
    return phrase

def remove_code(text):
    '''This function will remove code part and remove all html tags and return␣
 ↪lower case text'''
    text = text.lower()
    # make it proper spaced text
    text = ' '.join(text.split())
    # remove code area from text and replace with space
    text = re.sub('<code>.*?</code>',' ',text)
    # remove all tags which are in "< >"
    text = re.sub('<.*?>',' ',text)
    # remove content between img tags
    text = re.sub('<img.*?>.*?</img>',' ',text)
    # remove content between a tag and URLs
    text = re.sub('<a.*?>.*?</a>',' ',text)
    # decontract text
    text = decontracted(text)
    # remove \n from text
    text = text.replace('\n',' ')
    # remove punctuations except c# and c++
    text = re.sub('[^A-Za-zc#c++]+',' ',text)
    return text.lower()

def remove_stopwords(text):
    '''this function will remove stopwords from text using nltk stopwords'''
    final_text = ''
    for word in text.split():
        if word not in stopwords.words('english'):
            final_text += word + ' '
```

```
    return final_text

def preprocess_text(text):
    '''this function will do all preprocessing'''
    # remove code part and punctuations
    removed_code = remove_code(text)
    # remove stopwords
    preprocessed_text = remove_stopwords(removed_code)
    # return cleaned text
    return preprocessed_text

def preprocess_title(text):
    # convert to lower case
    text = text.lower()
    # decontract
    text = decontracted(text)
    # remove all punctuations except a-z and c# and c++
    text = re.sub('[^a-zc#c++]+',' ',text)
    # remove stop words
    text = remove_stopwords(text)
    return text
```

- In Question body and answer body we have to remove code parts and we have to get only text part so we will remove code part and will remove html tags and all punctuations and we will get clean text in lower case.
- We will add new column as preprocessed question body and preprocessed answer body.

[ ]:
```
# remove code part from question body
removed_code_questions = []
for i,row in tqdm(df.iterrows()):
    removed_code_questions.append(remove_code(row.body))
```

```
HBox(children=(FloatProgress(value=1.0, bar_style='info', max=1.0), HTML(value='')))
```

[ ]:
```
# remove stopwords from removed_code_questions
preprocessed_questions = []
for i in tqdm(removed_code_questions):
    preprocessed_questions.append(remove_stopwords(i))
```

```
100%|      | 572554/572554 [1:13:31<00:00, 129.79it/s]
```

[ ]:
```
# preprocess title
preprocessed_title = []
for i,row in tqdm(df.iterrows()):
    preprocessed_title.append(preprocess_title(row.title))
```

```
572554it [08:45, 1089.70it/s]
```

[ ]:
```
# original title
df['title'][2]
```

[ ]:
```
'How would you access Object properties from within an object method?'
```

[ ]:
```
# cleaned title
preprocessed_title[2]
```

[ ]:
```
'would access object properties within object method '
```

[ ]:
```
# remove code from answers
removed_code_answers = []
for row in tqdm(df['answers']):
    removed_code_answers.append(remove_code(row))
```

```
100%|      | 572554/572554 [00:46<00:00, 12388.69it/s]
```

[ ]:
```
# original answer at index 0
df['answers'][0]
```

[ ]:
```
'<p>The following approach (extract from <a
href="https://www.codeproject.com/Articles/168662/Time-Period-Library-for-NET"
rel="nofollow noreferrer">Time Period Library for .NET</a> class
<em>DateDiff</em>) considers the calendar of the culture
info:</p>\n\n<pre><code>//
-------------------------------------------------------------------------\nprivate
static int YearDiff( DateTime date1, DateTime date2 )\n{\n  return YearDiff(
date1, date2, DateTimeFormatInfo.CurrentInfo.Calendar );\n} // YearDiff\n\n//
-------------------------------------------------------------------------\nprivate
static int YearDiff( DateTime date1, DateTime date2, Calendar calendar )\n{\n
if ( date1.Equals( date2 ) )\n  {\n    return 0;\n  }\n\n  int year1 =
calendar.GetYear( date1 );\n  int month1 = calendar.GetMonth( date1 );\n  int
year2 = calendar.GetYear( date2 );\n  int month2 = calendar.GetMonth( date2
);\n\n  // find the the day to compare\n  int compareDay = date2.Day;\n  int
compareDaysPerMonth = calendar.GetDaysInMonth( year1, month1 );\n  if (
compareDay &gt; compareDaysPerMonth )\n  {\n    compareDay =
compareDaysPerMonth;\n  }\n\n  // build the compare date\n  DateTime compareDate
= new DateTime( year1, month2, compareDay,\n    date2.Hour, date2.Minute,
date2.Second, date2.Millisecond );\n  if ( date2 &gt; date1 )\n  {\n    if (
compareDate &lt; date1 )\n    {\n      compareDate = compareDate.AddYears( 1
);\n    }\n  }\n  else\n  {\n    if ( compareDate &gt; date1 )\n    {\n
compareDate = compareDate.AddYears( -1 );\n    }\n  }\n  return year2 -
calendar.GetYear( compareDate );\n} //
YearDiff\n</code></pre>\n\n<p>Usage:</p>\n\n<pre><code>//
-------------------------------------------------------------------------\npublic
void CalculateAgeSamples()\n{\n  PrintAge( new DateTime( 2000, 02, 29 ), new
```

DateTime(2009, 02, 28));\n // > Birthdate=29.02.2000, Age at 28.02.2009 is 8 years\n PrintAge(new DateTime(2000, 02, 29), new DateTime(2012, 02, 28));\n // > Birthdate=29.02.2000, Age at 28.02.2012 is 11 years\n} // CalculateAgeSamples\n\n//
----------\npublic void PrintAge(DateTime birthDate, DateTime moment)\n{\n Console.WriteLine("Birthdate={0:d}, Age at {1:d} is {2} years", birthDate, moment, YearDiff(birthDate, moment));\n} // PrintAge\n</code></pre>\n Here is a solution. \n\n<pre><code>DateTime dateOfBirth = new DateTime(2000, 4, 18);\nDateTime currentDate = DateTime.Now;\n\nint ageInYears = 0;\nint ageInMonths = 0;\nint ageInDays = 0;\n\nageInDays = currentDate.Day - dateOfBirth.Day;\nageInMonths = currentDate.Month - dateOfBirth.Month;\nageInYears = currentDate.Year - dateOfBirth.Year;\n\nif (ageInDays < 0)\n{\n   ageInDays += DateTime.DaysInMonth(currentDate.Year, currentDate.Month);\n   ageInMonths = ageInMonths--;\n\n   if (ageInMonths < 0)\n   {\n   ageInMonths += 12;\n   ageInYears--;\n   }\n}\n\nif (ageInMonths < 0)\n{\n   ageInMonths += 12;\n   ageInYears--;\n}\n\nConsole.WriteLine("{0}, {1}, {2}", ageInYears, ageInMonths, ageInDays);\n</code></pre>\n My suggestion \n\n<pre class="lang-cs prettyprint-override"><code>int age = (int) ((DateTime.Now - bday).TotalDays/365.242199);\n</code></pre>\n\n That seems to have the year changing on the right date. (I spot tested up to age 107.) \n Very simple answer \n\n<pre><code>   DateTime dob = new DateTime(1991, 3, 4); \n DateTime now = DateTime.Now; \n   int dobDay = dob.Day, dobMonth = dob.Month; \n   int add = -1; \n   if (dobMonth < now.Month)\n {\n   add = 0;\n   }\n   else if (dobMonth == now.Month)\n {\n   if(dobDay <= now.Day)\n   {\n   add = 0;\n   }\n   else\n   {\n   add = -1;\n }\n   }\n   else\n   {\n   add = -1;\n   } \n int age = now.Year - dob.Year + add;\n</code></pre>\n Here is a very simple and easy to follow example. \n\n<pre><code>private int CalculateAge()\n{\n//get birthdate\n   DateTime dtBirth = Convert.ToDateTime(BirthDatePicker.Value);\n   int byear = dtBirth.Year;\n   int bmonth = dtBirth.Month;\n   int bday = dtBirth.Day;\n   DateTime dtToday = DateTime.Now;\n   int tYear = dtToday.Year;\n   int tmonth = dtToday.Month;\n int tday = dtToday.Day;\n   int age = tYear - byear;\n   if (bmonth < tmonth)\n   age--;\n   else if (bmonth == tmonth && bday>tday)\n {\n   age--;\n   }\nreturn age;\n}\n</code></pre>\n Just use: \n\n<pre class="lang-cs prettyprint-override"><code>(DateTime.Now - myDate).TotalHours / 8766.0\n</code></pre>\n\n The current date - <code>myDate = TimeSpan</code>, get total hours and divide in the total hours per year and get exactly the age/months/days… \n This may work: \n\n<pre class="lang-cs prettyprint-override"><code>public override bool IsValid(DateTime value)\n{\n _dateOfBirth = value;\n   var yearsOld = (double) (DateTime.Now.Subtract(_dateOfBirth).TotalDays/365);\n   if (yearsOld > 18)\n   return true;\n   return false;\n}\n</code></pre>\n Here\'s a DateTime extender that adds the age calculation to the DateTime

object. \n\n<pre class="lang-cs prettyprint-override"><code>public static class AgeExtender\n{\n   public static int GetAge(this DateTime dt)\n   {\n int d = int.Parse(dt.ToString("yyyyMMdd"));\n   int t = int.Parse(DateTime.Today.ToString("yyyyMMdd"));\n   return (t-d)/10000;\n }\n}\n</code></pre>\n This is one of the most accurate answers that is able to resolve the birthday of 29th of Feb compared to any year of 28th Feb. \n\n<pre class="lang-cs prettyprint-override"><code>public int GetAge(DateTime birthDate)\n{\n   int age = DateTime.Now.Year - birthDate.Year;\n\n   if (birthDate.DayOfYear > DateTime.Now.DayOfYear)\n age--;\n\n   return age;\n}\n\n\n\n\n</code></pre>\n Do we need to consider people who is smaller than 1 year? as Chinese culture, we describe small babies\' age as 2 months or 4 weeks. \n\n Below is my implementation, it is not as simple as what I imagined, especially to deal with date like 2/28. \n\n<pre><code>public static string HowOld(DateTime birthday, DateTime now)\n{\n   if (now < birthday)\n   throw new ArgumentOutOfRangeException("birthday must be less than now.");\n\n   TimeSpan diff = now - birthday;\n   int diffDays = (int)diff.TotalDays;\n\n   if (diffDays > 7)//year, month and week\n   {\n   int age = now.Year - birthday.Year;\n\n   if (birthday > now.AddYears(-age))\n age--;\n\n   if (age > 0)\n   {\n   return age + (age > 1 ? " years" : " year");\n   }\n   else\n   {// month and week\n DateTime d = birthday;\n   int diffMonth = 1;\n\n   while (d.AddMonths(diffMonth) <= now)\n   {\n diffMonth++;\n   }\n\n   age = diffMonth-1;\n\n   if (age == 1 && d.Day > now.Day)\n   age--;\n\n if (age > 0)\n   {\n   return age + (age > 1 ? " months" : " month");\n   }\n   else\n   {\n age = diffDays / 7;\n   return age + (age > 1 ? " weeks" : " week");\n   }\n   }\n   }\n   else if (diffDays > 0)\n   {\n int age = diffDays;\n   return age + (age > 1 ? " days" : " day");\n }\n   else\n   {\n   int age = diffDays;\n   return "just born";\n }\n}\n</code></pre>\n\n This implementation has passed below test cases. \n\n<pre><code>[TestMethod]\npublic void TestAge()\n{\n   string age = HowOld(new DateTime(2011, 1, 1), new DateTime(2012, 11, 30));\n Assert.AreEqual("1 year", age);\n\n   age = HowOld(new DateTime(2011, 11, 30), new DateTime(2012, 11, 30));\n   Assert.AreEqual("1 year", age);\n\n   age = HowOld(new DateTime(2001, 1, 1), new DateTime(2012, 11, 30));\n Assert.AreEqual("11 years", age);\n\n   age = HowOld(new DateTime(2012, 1, 1), new DateTime(2012, 11, 30));\n   Assert.AreEqual("10 months", age);\n\n   age = HowOld(new DateTime(2011, 12, 1), new DateTime(2012, 11, 30));\n Assert.AreEqual("11 months", age);\n\n   age = HowOld(new DateTime(2012, 10, 1), new DateTime(2012, 11, 30));\n   Assert.AreEqual("1 month", age);\n\n age = HowOld(new DateTime(2008, 2, 28), new DateTime(2009, 2, 28));\n Assert.AreEqual("1 year", age);\n\n   age = HowOld(new DateTime(2008, 3, 28), new DateTime(2009, 2, 28));\n   Assert.AreEqual("11 months", age);\n\n   age = HowOld(new DateTime(2008, 3, 28), new DateTime(2009, 3, 28));\n Assert.AreEqual("1 year", age);\n\n   age = HowOld(new DateTime(2009, 1, 28),

new DateTime(2009, 2, 28));\n   Assert.AreEqual("1 month", age);\n\n   age = HowOld(new DateTime(2009, 2, 1), new DateTime(2009, 3, 1));\n Assert.AreEqual("1 month", age);\n\n   // NOTE.\n   // new DateTime(2008, 1, 31).AddMonths(1) == new DateTime(2009, 2, 28);\n   // new DateTime(2008, 1, 28).AddMonths(1) == new DateTime(2009, 2, 28);\n   age = HowOld(new DateTime(2009, 1, 31), new DateTime(2009, 2, 28));\n   Assert.AreEqual("4 weeks", age);\n\n   age = HowOld(new DateTime(2009, 2, 1), new DateTime(2009, 2, 28));\n   Assert.AreEqual("3 weeks", age);\n\n   age = HowOld(new DateTime(2009, 2, 1), new DateTime(2009, 3, 1));\n   Assert.AreEqual("1 month", age);\n\n   age = HowOld(new DateTime(2012, 11, 5), new DateTime(2012, 11, 30));\n   Assert.AreEqual("3 weeks", age);\n\n   age = HowOld(new DateTime(2012, 11, 1), new DateTime(2012, 11, 30));\n   Assert.AreEqual("4 weeks", age);\n\n   age = HowOld(new DateTime(2012, 11, 20), new DateTime(2012, 11, 30));\n   Assert.AreEqual("1 week", age);\n\n   age = HowOld(new DateTime(2012, 11, 25), new DateTime(2012, 11, 30));\n   Assert.AreEqual("5 days", age);\n\n   age = HowOld(new DateTime(2012, 11, 29), new DateTime(2012, 11, 30));\n   Assert.AreEqual("1 day", age);\n\n   age = HowOld(new DateTime(2012, 11, 30), new DateTime(2012, 11, 30));\n   Assert.AreEqual("just born", age);\n\n   age = HowOld(new DateTime(2000, 2, 29), new DateTime(2009, 2, 28));\n   Assert.AreEqual("8 years", age);\n\n   age = HowOld(new DateTime(2000, 2, 29), new DateTime(2009, 3, 1));\n   Assert.AreEqual("9 years", age);\n\n   Exception e = null;\n\n   try\n   {\n   age = HowOld(new DateTime(2012, 12, 1), new DateTime(2012, 11, 30));\n   }\n   catch (ArgumentOutOfRangeException ex)\n   {\n   e = ex;\n   }\n\n Assert.IsTrue(e != null);\n}\n</code></pre>\n\n Hope it\'s helpful. \n Just because I don\'t think the top answer is that clear: \n\n<pre class="lang-cs prettyprint-override"><code>public static int GetAgeByLoop(DateTime birthday)\n{\n   var age = -1;\n\n   for (var date = birthday; date < DateTime.Today; date = date.AddYears(1))\n   {\n age++;\n   }\n\n   return age;\n}\n</code></pre>\n I often count on my fingers. I need to look at a calendar to work out when things change. So that\'s what I\'d do in my code: \n\n<pre><code>int AgeNow(DateTime birthday)\n{\n return AgeAt(DateTime.Now, birthday);\n}\n\nint AgeAt(DateTime now, DateTime birthday)\n{\n   return AgeAt(now, birthday, CultureInfo.CurrentCulture.Calendar);\n}\n\nint AgeAt(DateTime now, DateTime birthday, Calendar calendar)\n{\n   // My age has increased on the morning of my\n   // birthday even though I was born in the evening.\n   now = now.Date;\n   birthday = birthday.Date;\n\n   var age = 0;\n   if (now <= birthday) return age; // I am zero now if I am to be born tomorrow.\n\n   while (calendar.AddYears(birthday, age + 1) <= now)\n   {\n   age++;\n   }\n return age;\n}\n</code></pre>\n\n Running this through in LINQPad gives this: \n\n<pre><code>PASSED: someone born on 28 February 1964 is age 4 on 28 February 1968\nPASSED: someone born on 29 February 1964 is age 3 on 28 February 1968\nPASSED: someone born on 31 December 2016 is age 0 on 01 January 2017\n</code></pre>\n\n Code in LINQPad is <a href="http://share.linqpad.net/3gx8ba.linq" rel="nofollow

noreferrer">here. \n This is the easiest way to answer this in a single line. \n\n<pre class="lang-cs prettyprint-override"><code>DateTime Dob = DateTime.Parse("1985-04-24");\n\nint Age = DateTime.MinValue.AddDays(DateTime.Now.Subtract(Dob).TotalHours/24).Year - 1;\n</code></pre>\n\n This also works for leap years. \n<pre class="lang-cs prettyprint-override"><code>private int GetYearDiff(DateTime start, DateTime end)\n{\n   int diff = end.Year - start.Year;\n   if (end.DayOfYear < start.DayOfYear) { diff -= 1; }\n   return diff;\n}\n[Fact]\npublic void GetYearDiff_WhenCalls_ShouldReturnCorrectYearDiff()\n{\n   //arrange\n   var now = DateTime.Now;\n   //act\n   //assert\n   Assert.Equal(24, GetYearDiff(new DateTime(1992, 7, 9), now)); // passed\n   Assert.Equal(24, GetYearDiff(new DateTime(1992, now.Month, now.Day), now)); // passed\n Assert.Equal(23, GetYearDiff(new DateTime(1992, 12, 9), now)); // passed\n}\n</code></pre>\n Wow, I had to give my answer here… There are so many answers for such a simple question. \n\n<pre><code>private int CalcularIdade(DateTime dtNascimento)\n   {\n   var nHoje = Convert.ToInt32(DateTime.Today.ToString("yyyyMMdd"));\n   var nAniversario = Convert.ToInt32(dtNascimento.ToString("yyyyMMdd"));\n\n   double diff = (nHoje - nAniversario) / 10000;\n\n   var ret = Convert.ToInt32(Math.Truncate(diff));\n\n   return ret;\n }\n</code></pre>\n I\'ve made one small change to Mark Soen\'s answer: I\'ve rewriten the third line so that the expression can be parsed a bit more easily. \n\n<pre><code>public int AgeInYears(DateTime bday)\n{\n   DateTime now = DateTime.Today;\n   int age = now.Year - bday.Year;   \n   if (bday.AddYears(age) > now) \n age--;\n   return age;\n}\n</code></pre>\n\n I\'ve also made it into a function for the sake of clarity. \n I\'ve spent some time working on this and came up with this to calculate someone\'s age in years, months and days. I\'ve tested against the Feb 29th problem and leap years and it seems to work, I\'d appreciate any feedback: \n\n<pre><code>public void LoopAge(DateTime myDOB, DateTime FutureDate)\n{\n   int years = 0;\n   int months = 0;\n   int days = 0;\n\n   DateTime tmpMyDOB = new DateTime(myDOB.Year, myDOB.Month, 1);\n\n   DateTime tmpFutureDate = new DateTime(FutureDate.Year, FutureDate.Month, 1);\n\n   while (tmpMyDOB.AddYears(years).AddMonths(months) < tmpFutureDate)\n   {\n   months++;\n\n   if (months > 12)\n {\n   years++;\n   months = months - 12;\n   }\n   }\n\n if (FutureDate.Day >= myDOB.Day)\n   {\n   days = days + FutureDate.Day - myDOB.Day;\n   }\n   else\n   {\n   months--;\n\n   if (months < 0)\n   {\n   years--;\n   months = months + 12;\n }\n\n   days +=\n   DateTime.DaysInMonth(\n FutureDate.AddMonths(-1).Year, FutureDate.AddMonths(-1).Month\n   ) + FutureDate.Day - myDOB.Day;\n\n   }\n\n   //add an extra day if the dob is a leap day\n   if (DateTime.IsLeapYear(myDOB.Year) && myDOB.Month == 2 && myDOB.Day == 29)\n   {\n   //but only if the future date is less than 1st March\n   if (FutureDate >= new DateTime(FutureDate.Year, 3, 1))\n   days++;\n   }\n\n}\n</code></pre>\n This classic question

is deserving of a Noda Time solution. \n\n<pre><code>static int GetAge(LocalDate dateOfBirth)\n{\n Instant now = SystemClock.Instance.Now;\n\n   // The target time zone is important.\n   // It should align with the *current physical location* of the person\n   // you are talking about. When the whereabouts of that person are unknown,\n   // then you use the time zone of the person who is *asking* for the age.\n   // The time zone of birth is irrelevant!\n\n   DateTimeZone zone = DateTimeZoneProviders.Tzdb["America/New_York"];\n\n   LocalDate today = now.InZone(zone).Date;\n\n   Period period = Period.Between(dateOfBirth, today, PeriodUnits.Years);\n\n   return (int) period.Years;\n}\n</code></pre>\n\n Usage: \n\n<pre><code>LocalDate dateOfBirth = new LocalDate(1976, 8, 27);\nint age = GetAge(dateOfBirth);\n</code></pre>\n\n You might also be interested in the following improvements: \n\n
\n Passing in the clock as an <code>IClock</code>, instead of using <code>SystemClock.Instance</code>, would improve testability. \n The target time zone will likely change, so you\'d want a <code>DateTimeZone</code> parameter as well. \n
\n\n See also my blog post on this subject: Handling Birthdays, and Other Anniversaries \n An easy to understand and simple solution. \n\n<pre class="lang-cs prettyprint-override"><code>\n// Save today\'s date.\nvar today = DateTime.Today;\n\n// Calculate the age.\nvar age = today.Year - birthdate.Year;\n\n// Go back to the year the person was born in case of a leap year\nif (birthdate.Date > today.AddYears(-age)) age--;\n</code></pre>\n\n However, this assumes you are looking for the western idea of age and not using East Asian reckoning. \n A one-liner answer: \n\n<pre class="lang-cs prettyprint-override"><code>DateTime dateOfBirth = Convert.ToDateTime("01/16/1990");\nvar age = ((DateTime.Now - dateOfBirth).Days) / 365;\n</code></pre>\n To calculate how many years old a person is, \n\n<pre class="lang-cs prettyprint-override"><code>DateTime dateOfBirth;\n\nint ageInYears = DateTime.Now.Year - dateOfBirth.Year;\n\nif (dateOfBirth > today.AddYears(-ageInYears)) ageInYears --;\n</code></pre>\n Here\'s a little code sample for C# I knocked up, be careful around the edge cases specifically leap years, not all the above solutions take them into account. Pushing the answer out as a DateTime can cause problems as you could end up trying to put too many days into a specific month e.g. 30 days in Feb. \n\n<pre><code>public string LoopAge(DateTime myDOB, DateTime FutureDate)\n{\n   int years = 0;\n   int months = 0;\n   int days = 0;\n\n   DateTime tmpMyDOB = new DateTime(myDOB.Year, myDOB.Month, 1);\n\n DateTime tmpFutureDate = new DateTime(FutureDate.Year, FutureDate.Month, 1);\n\n while (tmpMyDOB.AddYears(years).AddMonths(months) < tmpFutureDate)\n   {\n months++;\n   if (months > 12)\n   {\n   years++;\n months = months - 12;\n   }\n   }\n\n   if (FutureDate.Day >= myDOB.Day)\n   {\n   days = days + FutureDate.Day - myDOB.Day;\n   }\n

else\n   {\n   months--;\n   if (months < 0)\n   {\n
years--;\n   months = months + 12;\n   }\n   days = days +
(DateTime.DaysInMonth(FutureDate.AddMonths(-1).Year,
FutureDate.AddMonths(-1).Month) + FutureDate.Day) - myDOB.Day;\n\n   }\n\n
//add an extra day if the dob is a leap day\n   if
(DateTime.IsLeapYear(myDOB.Year) && myDOB.Month == 2 &&
myDOB.Day == 29)\n   {\n   //but only if the future date is less than 1st
March\n   if(FutureDate >= new DateTime(FutureDate.Year, 3,1))\n
days++;\n   }\n\n   return "Years: " + years + " Months: " + months + " Days:
" + days;\n}\n</code></pre>\n This is simple and appears to be accurate for my
needs. I am making an assumption for the purpose of leap years that regardless
of when the person chooses to celebrate the birthday they are not technically a
year older until 365 days have passed since their last birthday (i.e 28th
February does not make them a year older). \n<pre><code>DateTime now =
DateTime.Today;\nDateTime birthday = new DateTime(1991, 02, 03);//3rd feb\n\nint
age = now.Year - birthday.Year;\n\nif (now.Month < birthday.Month ||
(now.Month == birthday.Month && now.Day < birthday.Day))//not had
bday this year yet\n age--;\n\nreturn age;\n</code></pre>\n Let us know if
you spot any problems ;) \n Check this out: \n\n<pre class="lang-cs
prettyprint-override"><code>TimeSpan ts = DateTime.Now.Subtract(Birthdate);\nage
= (byte)(ts.TotalDays / 365.25);\n</code></pre>\n The simple answer to this is
to apply <code>AddYears</code> as shown below because this is the only native
method to add years to the 29th of Feb. of leap years and obtain the correct
result of the 28th of Feb. for common years. \n\n Some feel that 1th of
Mar. is the birthday of leaplings but neither .Net nor any official rule
supports this, nor does common logic explain why some born in February should
have 75% of their birthdays in another month. \n\n Further, an Age method
lends itself to be added as an extension to <code>DateTime</code>. By this you
can obtain the age in the simplest possible way: \n\n<ol>\n<li>List
item
\n
\n\n int age =
birthDate.Age(); \n\n<pre><code>public static class
DateTimeExtensions\n{\n   /// <summary>\n   /// Calculates the age in
years of the current System.DateTime object today.\n   /// </summary>\n
/// <param name="birthDate">The date of birth</param>\n   ///
<returns>Age in years today. 0 is returned for a future date of
birth.</returns>\n   public static int Age(this DateTime birthDate)\n
{\n   return Age(birthDate, DateTime.Today);\n   }\n\n   ///
<summary>\n   /// Calculates the age in years of the current
System.DateTime object on a later date.\n   /// </summary>\n   ///
<param name="birthDate">The date of birth</param>\n   /// <param
name="laterDate">The date on which to calculate the age.</param>\n
/// <returns>Age in years on a later day. 0 is returned as
minimum.</returns>\n   public static int Age(this DateTime birthDate,
DateTime laterDate)\n   {\n   int age;\n   age = laterDate.Year -
birthDate.Year;\n\n   if (age > 0)\n   {\n   age -=
Convert.ToInt32(laterDate.Date < birthDate.Date.AddYears(age));\n   }\n
else\n   {\n   age = 0;\n   }\n\n   return age;\n

}\n}\n</code></pre>\n\n Now, run this test: \n\n<pre><code>class Program\n{\n   static void Main(string[] args)\n   {\n   RunTest();\n }\n\n   private static void RunTest()\n   {\n   DateTime birthDate = new DateTime(2000, 2, 28);\n   DateTime laterDate = new DateTime(2011, 2, 27);\n   string iso = "yyyy-MM-dd";\n\n   for (int i = 0; i < 3; i++)\n   {\n   for (int j = 0; j < 3; j++)\n   {\n Console.WriteLine("Birth date: " + birthDate.AddDays(i).ToString(iso) + " Later date: " + laterDate.AddDays(j).ToString(iso) + " Age: " + birthDate.AddDays(i).Age(laterDate.AddDays(j)).ToString());\n   }\n }\n\n   Console.ReadKey();\n   }\n}\n</code></pre>\n\n The critical date example is this: \n\n Birth date: 2000-02-29 Later date: 2011-02-28 Age: 11 \n\n Output: \n\n<pre><code>{\n   Birth date: 2000-02-28 Later date: 2011-02-27 Age: 10\n   Birth date: 2000-02-28 Later date: 2011-02-28 Age: 11\n   Birth date: 2000-02-28 Later date: 2011-03-01 Age: 11\n   Birth date: 2000-02-29 Later date: 2011-02-27 Age: 10\n   Birth date: 2000-02-29 Later date: 2011-02-28 Age: 11\n   Birth date: 2000-02-29 Later date: 2011-03-01 Age: 11\n   Birth date: 2000-03-01 Later date: 2011-02-27 Age: 10\n   Birth date: 2000-03-01 Later date: 2011-02-28 Age: 10\n   Birth date: 2000-03-01 Later date: 2011-03-01 Age: 11\n}\n</code></pre>\n\n And for the later date 2012-02-28: \n\n<pre><code>{\n   Birth date: 2000-02-28 Later date: 2012-02-28 Age: 12\n   Birth date: 2000-02-28 Later date: 2012-02-29 Age: 12\n   Birth date: 2000-02-28 Later date: 2012-03-01 Age: 12\n   Birth date: 2000-02-29 Later date: 2012-02-28 Age: 11\n   Birth date: 2000-02-29 Later date: 2012-02-29 Age: 12\n   Birth date: 2000-02-29 Later date: 2012-03-01 Age: 12\n   Birth date: 2000-03-01 Later date: 2012-02-28 Age: 11\n   Birth date: 2000-03-01 Later date: 2012-02-29 Age: 11\n   Birth date: 2000-03-01 Later date: 2012-03-01 Age: 12\n}\n</code></pre>\n\n 2 Main problems to solve are: \n\n 1. Calculate Exact age - in years, months, days, etc. \n\n 2. Calculate Generally perceived age - people usually do not care how old they exactly are, they just care when their birthday in the current year is. \n\n<hr>\n\n Solution for 1 is obvious: \n\n<pre><code>DateTime birth = DateTime.Parse("1.1.2000");\nDateTime today = DateTime.Today;   //we usually don\'t care about birth time\nTimeSpan age = today - birth;   //.NET FCL should guarantee this as precise\ndouble ageInDays = age.TotalDays;   //total number of days … also precise\ndouble daysInYear = 365.2425;   //statistical value for 400 years\ndouble ageInYears = ageInDays / daysInYear;   //can be shifted … not so precise\n</code></pre>\n\n<hr>\n\n Solution for 2 is the one which is not so precise in determing total age, but is perceived as precise by people. People also usually use it, when they calculate their age "manually": \n\n<pre><code>DateTime birth = DateTime.Parse("1.1.2000");\nDateTime today = DateTime.Today;\nint age = today.Year - birth.Year;   //people perceive their age in years\n\nif (today.Month < birth.Month ||\n   ((today.Month == birth.Month) && (today.Day < birth.Day)))\n{\n age--; //birthday in current year not yet reached, we are 1 year younger ;)\n   //+ no birthday for 29.2. guys …

sorry, just wrong date for birth\n}\n</code></pre>\n\n Notes to 2.: \n\n
\n
This is my preferred solution
\n
We cannot use DateTime.DayOfYear or TimeSpans, as they shift number of days in leap years
\n
I have put there little more lines for readability
\n
\n\n Just one more note … I would create 2 static overloaded methods for it, one for universal usage, second for usage-friendliness: \n\n<pre><code>public static int GetAge(DateTime bithDay, DateTime today) \n{ \n //chosen solution method body\n}\n\npublic static int GetAge(DateTime birthDay) \n{ \n return GetAge(birthDay, DateTime.Now);\n}\n</code></pre>\n Here\'s a one-liner: \n\n<pre class="lang-cs prettyprint-override"><code>int age = new DateTime(DateTime.Now.Subtract(birthday).Ticks).Year-1;\n</code></pre>\n It can be this simple: \n\n<pre class="lang-cs prettyprint-override"><code>int age = DateTime.Now.AddTicks(0 - dob.Ticks).Year - 1;\n</code></pre>\n This is a strange way to do it, but if you format the date to <code>yyyymmdd</code> and subtract the date of birth from the current date then drop the last 4 digits you\'ve got the age :) \n\n I don\'t know C#, but I believe this will work in any language. \n\n<pre class="lang-cs prettyprint-override"><code>20080814 - 19800703 = 280111 \n</code></pre>\n\n Drop the last 4 digits = <code>28</code>. \n\n C# Code: \n\n<pre class="lang-cs prettyprint-override"><code>int now = int.Parse(DateTime.Now.ToString("yyyyMMdd"));\nint dob = int.Parse(dateOfBirth.ToString("yyyyMMdd"));\nint age = (now - dob) / 10000;\n</code></pre>\n\n Or alternatively without all the type conversion in the form of an extension method. Error checking omitted: \n\n<pre class="lang-cs prettyprint-override"><code>public static Int32 GetAge(this DateTime dateOfBirth)\n{\n   var today = DateTime.Today;\n\n   var a = (today.Year * 100 + today.Month) * 100 + today.Day;\n   var b = (dateOfBirth.Year * 100 + dateOfBirth.Month) * 100 + dateOfBirth.Day;\n\n return (a - b) / 10000;\n}\n</code></pre>\n This gives "more detail" to this question. Maybe this is what you\'re looking for \n\n<pre><code>DateTime birth = new DateTime(1974, 8, 29);\nDateTime today = DateTime.Now;\nTimeSpan span = today - birth;\nDateTime age = DateTime.MinValue + span;\n\n// Make adjustment due to MinValue equalling 1/1/1\nint years = age.Year - 1;\nint months = age.Month - 1;\nint days = age.Day - 1;\n\n// Print out not only how many years old they are but give months and days as well\nConsole.Write("{0} years, {1} months, {2} days", years, months, days);\n</code></pre>\n I have a customized method to calculate age, plus a bonus validation message just in case it helps: \n\n<pre><code>public void GetAge(DateTime dob, DateTime now, out int years, out int months, out int days)\n{\n   years = 0;\n   months = 0;\n   days = 0;\n\n   DateTime tmpdob = new DateTime(dob.Year, dob.Month, 1);\n   DateTime tmpnow = new DateTime(now.Year, now.Month, 1);\n\n   while (tmpdob.AddYears(years).AddMonths(months) < tmpnow)\n   {\n   months++;\n   if (months > 12)\n   {\n   years++;\n   months = months - 12;\n   }\n   }\n\n   if (now.Day >= dob.Day)\n   days = days + now.Day - dob.Day;\n   else\n   {\n   months--;\n   if (months < 0)\n   {\n   years--;\n   months = months + 12;\n   }\n   days += DateTime.DaysInMonth(now.AddMonths(-1).Year,

now.AddMonths(-1).Month) + now.Day - dob.Day;\n   }\n\n   if (DateTime.IsLeapYear(dob.Year) && dob.Month == 2 && dob.Day == 29 && now >= new DateTime(now.Year, 3, 1))\n   days++;\n\n} \n\nprivate string ValidateDate(DateTime dob) //This method will validate the date\n{\n   int Years = 0; int Months = 0; int Days = 0;\n\n   GetAge(dob, DateTime.Now, out Years, out Months, out Days);\n\n   if (Years < 18)\n message = Years + " is too young. Please try again on your 18th birthday.";\n else if (Years >= 65)\n   message = Years + " is too old. Date of Birth must not be 65 or older.";\n   else\n   return null; //Denotes validation passed\n}\n</code></pre>\n\n Method call here and pass out datetime value (MM/dd/yyyy if server set to USA locale). Replace this with anything a messagebox or any container to display: \n\n<pre><code>DateTime dob = DateTime.Parse("03/10/1982"); \n\nstring message = ValidateDate(dob);\n\nlbldatemessage.Visible = !StringIsNullOrWhitespace(message);\nlbldatemessage.Text = message ?? ""; //Ternary if message is null then default to empty string\n</code></pre>\n\n Remember you can format the message any way you like. \n Try this solution, it\'s working. \n\n<pre class="lang-cs prettyprint-override"><code>int age = (Int32.Parse(DateTime.Today.ToString("yyyyMMdd")) - \n Int32.Parse(birthday.ToString("yyyyMMdd rawrrr"))) / 10000;\n</code></pre>\n I\'ve created an Age struct, which looks like this: \n\n<pre><code>public struct Age : IEquatable<Age>, IComparable<Age>\n{\n   private readonly int _years;\n   private readonly int _months;\n   private readonly int _days;\n\n   public int Years { get { return _years; } }\n   public int Months { get { return _months; } }\n public int Days { get { return _days; } }\n\n   public Age(int years, int months, int days) : this()\n   {\n   _years = years;\n   _months = months;\n   _days = days;\n   }\n\n   public static Age CalculateAge(DateTime dateOfBirth, DateTime date)\n   {\n   // Here is some logic that ressembles Mike\'s solution, although it\n   // also takes into account months & days.\n   // Ommitted for brevity.\n   return new Age (years, months, days);\n   }\n\n   // Ommited Equality, Comparable, GetHashCode, functionality for brevity.\n}\n</code></pre>\n Here is a function that is serving me well. No calculations , very simple. \n<pre><code>public static string ToAge(this DateTime dob, DateTime? toDate = null)\n   {\n if (!toDate.HasValue)\n   toDate = DateTime.Now;\n   var now = toDate.Value;\n\n   if (now.CompareTo(dob) < 0)\n   return "Future date";\n\n   int years = now.Year - dob.Year;\n int months = now.Month - dob.Month;\n   int days = now.Day - dob.Day;\n\n if (days < 0)\n   {\n   months--;\n   days = DateTime.DaysInMonth(dob.Year, dob.Month) - dob.Day + now.Day;\n   }\n\n if (months < 0)\n   {\n   years--;\n   months = 12 + months;\n   }\n\n\n   return string.Format("{0} year(s), {1} month(s), {2} days(s)",\n   years,\n   months,\n days);\n   }\n</code></pre>\n And here is a unit test: \n<pre><code>[Test]\n   public void ToAgeTests()\n   {\n   var date = new

```
DateTime(2000, 1, 1);\n        Assert.AreEqual(&quot;0 year(s), 0 month(s), 1
days(s)&quot;, new DateTime(1999, 12, 31).ToAge(date));\n
Assert.AreEqual(&quot;0 year(s), 0 month(s), 0 days(s)&quot;, new DateTime(2000,
1, 1).ToAge(date));\n        Assert.AreEqual(&quot;1 year(s), 0 month(s), 0
days(s)&quot;, new DateTime(1999, 1, 1).ToAge(date));\n
Assert.AreEqual(&quot;0 year(s), 11 month(s), 0 days(s)&quot;, new
DateTime(1999, 2, 1).ToAge(date));\n        Assert.AreEqual(&quot;0 year(s), 10
month(s), 25 days(s)&quot;, new DateTime(1999, 2, 4).ToAge(date));\n
Assert.AreEqual(&quot;0 year(s), 10 month(s), 1 days(s)&quot;, new
DateTime(1999, 2, 28).ToAge(date));\n\n        date = new DateTime(2000, 2,
15);\n        Assert.AreEqual(&quot;0 year(s), 0 month(s), 28 days(s)&quot;, new
DateTime(2000, 1, 18).ToAge(date));\n    }\n</code></pre>\n<p>I used
ScArcher2\'s solution for an accurate Year calculation of a persons age but I
needed to take it further and calculate their Months and Days along with the
Years.</p>\n\n<pre><code>    public static Dictionary&lt;string,int&gt;
CurrentAgeInYearsMonthsDays(DateTime? ndtBirthDate, DateTime? ndtReferralDate)\n
{\n
//--------------------------------------------------------------------\n
// Can\'t determine age if we don\'t have a dates.\n
//--------------------------------------------------------------------\n
if (ndtBirthDate == null) return null;\n        if (ndtReferralDate == null)
return null;\n\n        DateTime dtBirthDate =
Convert.ToDateTime(ndtBirthDate);\n        DateTime dtReferralDate =
Convert.ToDateTime(ndtReferralDate);\n\n
//--------------------------------------------------------------------\n
// Create our Variables\n
//--------------------------------------------------------------------\n
Dictionary&lt;string, int&gt; dYMD = new Dictionary&lt;string,int&gt;();\n
int iNowDate, iBirthDate, iYears, iMonths, iDays;\n        string sDif = "";\n\n
//--------------------------------------------------------------------\n
// Store off current date/time and DOB into local variables\n
//-------------------------------------------------------------------- \n
iNowDate = int.Parse(dtReferralDate.ToString("yyyyMMdd"));\n        iBirthDate =
int.Parse(dtBirthDate.ToString("yyyyMMdd"));\n\n
//--------------------------------------------------------------------\n
// Calculate Years\n
//--------------------------------------------------------------------\n
sDif = (iNowDate - iBirthDate).ToString();\n        iYears =
int.Parse(sDif.Substring(0, sDif.Length - 4));\n\n
//--------------------------------------------------------------------\n
// Store Years in Return Value\n
//--------------------------------------------------------------------\n
dYMD.Add("Years", iYears);\n\n
//--------------------------------------------------------------------\n
// Calculate Months\n
//--------------------------------------------------------------------\n
if (dtBirthDate.Month &gt; dtReferralDate.Month)\n            iMonths = 12 -
```

```
dtBirthDate.Month + dtReferralDate.Month - 1;\n        else\n            iMonths
= dtBirthDate.Month - dtReferralDate.Month;\n\n
//------------------------------------------------------------------------\n
// Store Months in Return Value\n
//------------------------------------------------------------------------\n
dYMD.Add("Months", iMonths);\n\n
//------------------------------------------------------------------------\n
// Calculate Remaining Days\n
//------------------------------------------------------------------------\n
if (dtBirthDate.Day &gt; dtReferralDate.Day)\n            //Logic: Figure out
the days in month previous to the current month, or the admitted month.\n
//       Subtract the birthday from the total days which will give us how many
days the person has lived since their birthdate day the previous month.\n
//       then take the referral date and simply add the number of days the
person has lived this month.\n\n            //If referral date is january, we
need to go back to the following year\'s December to get the days in that
month.\n            if (dtReferralDate.Month == 1)\n                iDays =
DateTime.DaysInMonth(dtReferralDate.Year - 1, 12) - dtBirthDate.Day +
dtReferralDate.Day;       \n            else\n                iDays =
DateTime.DaysInMonth(dtReferralDate.Year, dtReferralDate.Month - 1) -
dtBirthDate.Day + dtReferralDate.Day;       \n        else\n            iDays =
dtReferralDate.Day - dtBirthDate.Day;             \n\n
//------------------------------------------------------------------------\n
// Store Days in Return Value\n
//------------------------------------------------------------------------\n
dYMD.Add("Days", iDays);\n\n        return dYMD;\n}\n</code></pre>\n<p>This is
not a direct answer, but more of a philosophical reasoning about the problem at
hand from a quasi-scientific point of view.</p>\n\n<p>I would argue that the
question does not specify the unit nor culture in which to measure age, most
answers seem to assume an integer annual representation. The SI-unit for time is
<code>second</code>, ergo the correct generic answer should be (of course
assuming normalized <code>DateTime</code> and taking no regard whatsoever to
relativistic effects):</p>\n\n<pre><code>var lifeInSeconds = (DateTime.Now.Ticks
- then.Ticks)/TickFactor;\n</code></pre>\n\n<p>In the Christian way of
calculating age in years:</p>\n\n<pre><code>var then = … // Then, in this case
the birthday\nvar now = DateTime.UtcNow;\nint age = now.Year - then.Year;\nif
(now.AddYears(-age) &lt; then) age--;\n</code></pre>\n\n<p>In finance there is a
similar problem when calculating something often referred to as the <em>Day
Count Fraction</em>, which roughly is a number of years for a given period. And
the age issue is really a time measuring issue.</p>\n\n<p>Example for the
actual/actual (counting all days "correctly")
convention:</p>\n\n<pre><code>DateTime start, end = … // Whatever, assume
start is before end\n\ndouble startYearContribution = 1 - (double)
start.DayOfYear / (double) (DateTime.IsLeapYear(start.Year) ? 366 :
365);\ndouble endYearContribution = (double)end.DayOfYear /
(double)(DateTime.IsLeapYear(end.Year) ? 366 : 365);\ndouble middleContribution
= (double) (end.Year - start.Year - 1);\n\ndouble DCF = startYearContribution +
```

endYearContribution + middleContribution;\n</code></pre>\n\n Another quite common way to measure time generally is by "serializing" (the dude who named this date convention must seriously have been trippin\'): \n\n<pre><code>DateTime start, end = … // Whatever, assume start is before end\nint days = (end - start).Days;\n</code></pre>\n\n I wonder how long we have to go before a relativistic age in seconds becomes more useful than the rough approximation of earth-around-sun-cycles during one\'s lifetime so far :) Or in other words, when a period must be given a location or a function representing motion for itself to be valid :) \n How about this solution? \n\n<pre><code>static string CalcAge(DateTime birthDay)\n{\n DateTime currentDate = DateTime.Now;   \n   int approximateAge = currentDate.Year - birthDay.Year;\n   int daysToNextBirthDay = (birthDay.Month * 30 + birthDay.Day) - \n   (currentDate.Month * 30 + currentDate.Day) ;\n\n   if (approximateAge == 0 || approximateAge == 1)\n   {\n   int month = Math.Abs(daysToNextBirthDay / 30);\n   int days = Math.Abs(daysToNextBirthDay % 30);\n\n   if (month == 0)\n return "Your age is: " + daysToNextBirthDay + " days";\n\n   return "Your age is: " + month + " months and " + days + " days"; ;\n   }\n\n   if (daysToNextBirthDay > 0)\n   return "Your age is: " + --approximateAge + " Years";\n\n   return "Your age is: " + approximateAge + " Years"; ;\n}\n</code></pre>\n Here\'s yet another answer: \n<pre><code>public static int AgeInYears(DateTime birthday, DateTime today)\n{\n   return ((today.Year - birthday.Year) * 372 + (today.Month - birthday.Month) * 31 + (today.Day - birthday.Day)) / 372;\n}\n</code></pre>\n This has been extensively unit-tested. It does look a bit "magic". The number 372 is the number of days there would be in a year if every month had 31 days. \n The explanation of why it works (lifted from here) is: \n<blockquote>\n Let\'s set <code>Yn = DateTime.Now.Year, Yb = birthday.Year, Mn = DateTime.Now.Month, Mb = birthday.Month, Dn = DateTime.Now.Day, Db = birthday.Day</code> \n <code>age = Yn - Yb + (31*(Mn - Mb) + (Dn - Db)) / 372</code> \n We know that what we need is either <code>Yn-Yb</code> if the date has already been reached, <code>Yn-Yb-1</code> if it has not. \n a) If <code>Mn<Mb</code>, we have <code>-341 <= 31*(Mn-Mb) <= -31 and -30 <= Dn-Db <= 30</code> \n <code>-371 <= 31*(Mn - Mb) + (Dn - Db) <= -1</code> \n With integer division \n <code>(31*(Mn - Mb) + (Dn - Db)) / 372 = -1</code> \n b) If <code>Mn=Mb</code> and <code>Dn<Db</code>, we have <code>31*(Mn - Mb) = 0 and -30 <= Dn-Db <= -1</code> \n With integer division, again \n <code>(31*(Mn - Mb) + (Dn - Db)) / 372 = -1</code> \n c) If <code>Mn>Mb</code>, we have <code>31 <= 31*(Mn-Mb) <= 341 and -30 <= Dn-Db <= 30</code> \n <code>1 <= 31*(Mn - Mb) + (Dn - Db) <= 371</code> \n With integer division \n <code>(31*(Mn - Mb) + (Dn - Db)) / 372 = 0</code> \n d) If <code>Mn=Mb</code> and <code>Dn>Db</code>, we have <code>31*(Mn - Mb) = 0 and 1 <= Dn-Db <= 3</code>0 \n With integer division,

again \n <code>(31*(Mn - Mb) + (Dn - Db)) / 372 = 0</code> \n e) If <code>Mn=Mb</code> and <code>Dn=Db</code>, we have <code>31*(Mn - Mb) + Dn-Db = 0</code> \n and therefore <code>(31*(Mn - Mb) + (Dn - Db)) / 372 = 0</code> \n</blockquote>\n The best way that I know of because of leap years and everything is: \n\n<pre class="lang-cs prettyprint-override"><code>DateTime birthDate = new DateTime(2000,3,1);\nint age = (int)Math.Floor((DateTime.Now - birthDate).TotalDays / 365.25D);\n</code></pre>\n Another function, not by me but found on the web and refined it a bit: \n\n<pre><code>public static int GetAge(DateTime birthDate)\n{\n   DateTime n = DateTime.Now; // To avoid a race condition around midnight\n   int age = n.Year - birthDate.Year;\n\n   if (n.Month < birthDate.Month || (n.Month == birthDate.Month && n.Day < birthDate.Day))\n   age--;\n\n   return age;\n}\n</code></pre>\n\n Just two things that come into my mind: What about people from countries that do not use the Gregorian calendar? DateTime.Now is in the server-specific culture I think. I have absolutely zero knowledge about actually working with Asian calendars and I do not know if there is an easy way to convert dates between calendars, but just in case you\'re wondering about those Chinese guys from the year 4660 :-) \n Simple Code \n\n<pre><code> var birthYear=1993;\n var age = DateTime.Now.AddYears(-birthYear).Year;\n</code></pre>\n === Common Saying (from months to years old) === \n\n If you just for common use, here is the code as your information: \n\n<pre><code>DateTime today = DateTime.Today;\nDateTime bday = DateTime.Parse("2016-2-14");\nint age = today.Year - bday.Year;\nvar unit = "";\n\nif (bday > today.AddYears(-age))\n{\n   age--;\n}\nif (age == 0)   // Under one year old\n{\n   age = today.Month - bday.Month;\n\n   age = age <= 0 ? (12 + age) : age; // The next year before birthday\n\n   age = today.Day - bday.Day >= 0 ? age : --age; // Before the birthday.day\n\n   unit = "month";\n}\nelse {\n   unit = "year";\n}\n\nif (age > 1)\n{\n   unit = unit + "s";\n}\n</code></pre>\n\n The test result as below: \n\n<pre><code>The birthday: 2016-2-14\n\n2016-2-15 => age=0, unit=month;\n2016-5-13 => age=2, unit=months;\n2016-5-14 => age=3, unit=months; \n2016-6-13 => age=3, unit=months; \n2016-6-15 => age=4, unit=months; \n2017-1-13 => age=10, unit=months; \n2017-1-14 => age=11, unit=months; \n2017-2-13 => age=11, unit=months; \n2017-2-14 => age=1, unit=year; \n2017-2-15 => age=1, unit=year; \n2017-3-13 => age=1, unit=year;\n2018-1-13 => age=1, unit=year; \n2018-1-14 => age=1, unit=year; \n2018-2-13 => age=1, unit=year; \n2018-2-14 => age=2, unit=years; \n</code></pre>\n<blockquote>\n How come the MSDN help did not tell you that? It looks so obvious: \n</blockquote>\n\n<pre class="lang-cs prettyprint-override"><code>System.DateTime birthTime = AskTheUser(myUser); // :-)\nSystem.DateTime now = System.DateTime.Now;\nSystem.TimeSpan age = now - birthTime; // As simple as that\ndouble ageInDays = age.TotalDays; // Will you convert to whatever you want yourself?\n</code></pre>\n I have used the following for this issue. I know it\'s not very elegant, but it\'s working. \n\n<pre class="lang-cs prettyprint-override"><code>DateTime zeroTime = new DateTime(1, 1, 1);\nvar date1 = new DateTime(1983, 03, 04);\nvar

date2 = DateTime.Now;\nvar dif = date2 - date1;\nint years = (zeroTime + dif).Year - 1;\nLog.DebugFormat("Years -->{0}", years);\n</code></pre>\n<pre class="lang-cs prettyprint-override"><code>public string GetAge(this DateTime birthdate, string ageStrinFormat = null)\n{\n   var date = DateTime.Now.AddMonths(-birthdate.Month).AddDays(-birthdate.Day);\n   return string.Format(ageStrinFormat ?? "{0}/{1}/{2}",\n   (date.Year - birthdate.Year), date.Month, date.Day);\n}\n</code></pre>\n Keeping it simple (and possibly stupid:)). \n\n<pre class="lang-cs prettyprint-override"><code>DateTime birth = new DateTime(1975, 09, 27, 01, 00, 00, 00);\nTimeSpan ts = DateTime.Now - birth;\nConsole.WriteLine("You are approximately " + ts.TotalSeconds.ToString() + " seconds old.");\n</code></pre>\n To calculate the age with nearest age: \n\n<pre class="lang-cs prettyprint-override"><code>var ts = DateTime.Now - new DateTime(1988, 3, 19);\nvar age = Math.Round(ts.Days / 365.0);\n</code></pre>\n<pre><code>private int GetAge(int _year, int _month, int _day\n{\n   DateTime yourBirthDate= new DateTime(_year, _month, _day);\n\n   DateTime todaysDateTime = DateTime.Today;\n   int noOfYears = todaysDateTime.Year - yourBirthDate.Year;\n\n   if (DateTime.Now.Month < yourBirthDate.Month ||\n   (DateTime.Now.Month == yourBirthDate.Month && DateTime.Now.Day < yourBirthDate.Day))\n   {\n   noOfYears--;\n   }\n\n   return noOfYears;\n}\n</code></pre>\n This is the version we use here. It works, and it\'s fairly simple. It\'s the same idea as Jeff\'s but I think it\'s a little clearer because it separates out the logic for subtracting one, so it\'s a little easier to understand. \n\n<pre class="lang-cs prettyprint-override"><code>public static int GetAge(this DateTime dateOfBirth, DateTime dateAsAt)\n{\n   return dateAsAt.Year - dateOfBirth.Year - (dateOfBirth.DayOfYear < dateAsAt.DayOfYear ? 0 : 1);\n}\n</code></pre>\n\n You could expand the ternary operator to make it even clearer, if you think that sort of thing is unclear. \n\n Obviously this is done as an extension method on <code>DateTime</code>, but clearly you can grab that one line of code that does the work and put it anywhere. Here we have another overload of the Extension method that passes in <code>DateTime.Now</code>, just for completeness. \n With fewer conversions and UtcNow, this code can take care of someone born on the Feb 29 in a leap year: \n\n<pre><code>public int GetAge(DateTime DateOfBirth)\n{\n   var Now = DateTime.UtcNow;\n   return Now.Year - DateOfBirth.Year -\n   (\n   (\n   Now.Month > DateOfBirth.Month ||\n   (Now.Month == DateOfBirth.Month && Now.Day >= DateOfBirth.Day)\n   ) ? 0 : 1\n   );\n}\n</code></pre>\n Here is the simplest way to calculate someone\'s age.
\nCalculating someone\'s age is pretty straightforward, and here\'s how! In order for the code to work, you need a DateTime object called BirthDate containing the birthday. \n\n<pre><code>C#\n   // get the difference in years\n   int years = DateTime.Now.Year - BirthDate.Year; \n   // subtract another year if we\'re before the\n   // birth day in the current year\n   if (DateTime.Now.Month < BirthDate.Month || \n   (DateTime.Now.Month == BirthDate.Month && \n   DateTime.Now.Day < BirthDate.Day))

\n   years--;\n VB.NET\n   \' get the difference in years\n Dim years As Integer = DateTime.Now.Year - BirthDate.Year\n   \' subtract another year if we\'re before the\n   \' birth day in the current year\n If DateTime.Now.Month < BirthDate.Month Or (DateTime.Now.Month = BirthDate.Month And DateTime.Now.Day < BirthDate.Day) Then \n years = years - 1\n   End If\n</code></pre>\n I would simply do this: \n\n<pre class="lang-cs prettyprint-override"><code>DateTime birthDay = new DateTime(1990, 05, 23);\nDateTime age = DateTime.Now - birthDay;\n</code></pre>\n\n This way you can calculate the exact age of a person, down to the millisecond if you want. \n<pre class="lang-cs prettyprint-override"><code>var birthDate = … // DOB\nvar resultDate = DateTime.Now - birthDate;\n</code></pre>\n\n Using <code>resultDate</code> you can apply <code>TimeSpan</code> properties whatever you want to display it. \n I think the TimeSpan has all that we need in it, without having to resort to 365.25 (or any other approximation). Expanding on Aug\'s example: \n\n<pre class="lang-cs prettyprint-override"><code>DateTime myBD = new DateTime(1980, 10, 10);\nTimeSpan difference = DateTime.Now.Subtract(myBD);\n\ntextBox1.Text = difference.Years + " years " + difference.Months + " Months " + difference.Days + " days";\n</code></pre>\n SQL version: \n\n<pre><code>declare @dd smalldatetime = \'1980-04-01\'\ndeclare @age int = YEAR(GETDATE())-YEAR(@dd)\nif (@dd> DATEADD(YYYY, -@age, GETDATE())) set @age = @age -1\n\nprint @age \n</code></pre>\n Here is a test snippet: \n\n<pre class="lang-cs prettyprint-override"><code>DateTime bDay = new DateTime(2000, 2, 29);\nDateTime now = new DateTime(2009, 2, 28);\nMessageBox.Show(string.Format("Test {0} {1} {2}",\n   CalculateAgeWrong1(bDay, now),   // outputs 9\n CalculateAgeWrong2(bDay, now),   // outputs 9\n CalculateAgeCorrect(bDay, now),   // outputs 8\n CalculateAgeCorrect2(bDay, now))); // outputs 8\n</code></pre>\n\n Here you have the methods: \n\n<pre class="lang-cs prettyprint-override"><code>public int CalculateAgeWrong1(DateTime birthDate, DateTime now)\n{\n   return new DateTime(now.Subtract(birthDate).Ticks).Year - 1;\n}\n\npublic int CalculateAgeWrong2(DateTime birthDate, DateTime now)\n{\n   int age = now.Year - birthDate.Year;\n\n   if (now < birthDate.AddYears(age))\n age--;\n\n   return age;\n}\n\npublic int CalculateAgeCorrect(DateTime birthDate, DateTime now)\n{\n   int age = now.Year - birthDate.Year;\n\n   if (now.Month < birthDate.Month || (now.Month == birthDate.Month && now.Day < birthDate.Day))\n   age--;\n\n   return age;\n}\n\npublic int CalculateAgeCorrect2(DateTime birthDate, DateTime now)\n{\n   int age = now.Year - birthDate.Year;\n\n   // For leap years we need this\n   if (birthDate > now.AddYears(-age)) \n   age--;\n   // Don\'t use:\n   // if (birthDate.AddYears(age) > now) \n   //   age--;\n\n   return age;\n}\n</code></pre>\n I have created a SQL Server User Defined Function to calculate someone\'s age, given their birthdate. This is useful when you need it as part of a query: \n\n<pre><code>using System;\nusing System.Data;\nusing System.Data.Sql;\nusing System.Data.SqlClient;\nusing System.Data.SqlTypes;\nusing Microsoft.SqlServer.Server;\n\npublic partial class

UserDefinedFunctions\n{\n   [SqlFunction(DataAccess = DataAccessKind.Read)]\n public static SqlInt32 CalculateAge(string strBirthDate)\n   {\n DateTime dtBirthDate = new DateTime();\n   dtBirthDate = Convert.ToDateTime(strBirthDate);\n   DateTime dtToday = DateTime.Now;\n\n // get the difference in years\n   int years = dtToday.Year - dtBirthDate.Year;\n\n   // subtract another year if we\'re before the\n // birth day in the current year\n   if (dtToday.Month < dtBirthDate.Month || (dtToday.Month == dtBirthDate.Month && dtToday.Day < dtBirthDate.Day))\n   years=years-1;\n\n   int intCustomerAge = years;\n   return intCustomerAge;\n   }\n};\n</code></pre>\n<pre class="lang-cs prettyprint-override"><code>TimeSpan diff = DateTime.Now - birthdayDateTime;\nstring age = String.Format("{0:%y} years, {0:%M} months, {0:%d}, days old", diff);\n</code></pre>\n\n I\'m not sure how exactly you\'d like it returned to you, so I just made a readable string. \n The simplest way I\'ve ever found is this. It works correctly for the US and western europe locales. Can\'t speak to other locales, especially places like China. 4 extra compares, at most, following the initial computation of age. \n\n<pre><code>public int AgeInYears(DateTime birthDate, DateTime referenceDate)\n{\n Debug.Assert(referenceDate >= birthDate, \n "birth date must be on or prior to the reference date");\n\n DateTime birth = birthDate.Date;\n DateTime reference = referenceDate.Date;\n int years = (reference.Year - birth.Year);\n\n //\n // an offset of -1 is applied if the birth date has \n // not yet occurred in the current year.\n //\n if (reference.Month > birth.Month);\n else if (reference.Month < birth.Month) \n   --years;\n else // in birth month\n {\n   if (reference.Day < birth.Day)\n   --years;\n }\n\n return years;\n}\n</code></pre>\n\n I was looking over the answers to this and noticed that nobody has made reference to regulatory/legal implications of leap day births. For instance, per Wikipedia, if you\'re born on February 29th in various jurisdictions, you\'re non-leap year birthday varies: \n\n \n In the United Kingdom and Hong Kong: it\'s the ordinal day of the year, so the next day, March 1st is your birthday. \n In New Zealand: it\'s the previous day, February 28th for the purposes of driver licencing, and March 1st for other purposes. \n Taiwan: it\'s February 28th. \n \n\n And as near as I can tell, in the US, the statutes are silent on the matter, leaving it up to the common law and to how various regulatory bodies define things in their regulations. \n\n To that end, an improvement: \n\n<pre><code>public enum LeapDayRule\n{\n OrdinalDay   = 1 ,\n LastDayOfMonth = 2 ,\n}\n\nstatic int ComputeAgeInYears(DateTime birth, DateTime reference, LeapYearBirthdayRule ruleInEffect)\n{\n bool isLeapYearBirthday = CultureInfo.CurrentCulture.Calendar.IsLeapDay(birth.Year, birth.Month, birth.Day);\n DateTime cutoff;\n\n if (isLeapYearBirthday && !DateTime.IsLeapYear(reference.Year))\n {\n   switch (ruleInEffect)\n   {\n   case LeapDayRule.OrdinalDay:\n   cutoff = new DateTime(reference.Year, 1, 1)\n .AddDays(birth.DayOfYear - 1);\n   break;\n\n   case

```
LeapDayRule.LastDayOfMonth:\n        cutoff = new DateTime(reference.Year,
birth.Month, 1)\n                                .AddMonths(1)\n
.AddDays(-1);\n        break;\n\n      default:\n        throw new
InvalidOperationException();\n    }\n  }\n  else\n  {\n    cutoff = new
DateTime(reference.Year, birth.Month, birth.Day);\n  }\n\n  int age =
(reference.Year - birth.Year) + (reference &gt;= cutoff ? 0 : -1);\n  return age
&lt; 0 ? 0 : age;\n}\n</code></pre>\n\n<p>It should be noted that this code
assumes:</p>\n\n<ul>\n<li>A western (European) reckoning of age, and</li>\n<li>A
calendar, like the Gregorian calendar that inserts a single leap day at the end
of a month.</li>\n</ul>\n<p>I use this:</p>\n\n<pre><code>public static class
DateTimeExtensions\n{\n    public static int Age(this DateTime birthDate)\n
{\n        return Age(birthDate, DateTime.Now);\n    }\n\n    public static int
Age(this DateTime birthDate, DateTime offsetDate)\n    {\n        int
result=0;\n        result = offsetDate.Year - birthDate.Year;\n\n        if
(offsetDate.DayOfYear &lt; birthDate.DayOfYear)\n        {\n
result--;\n        }\n\n        return result;\n    }\n}\n</code></pre>\n<p>I
want to add Hebrew calendar calculations (or other System.Globalization calendar
can be used in the same way),  using rewrited functions from this
thread:</p>\n\n<pre class="lang-cs prettyprint-override"><code>Public Shared
Function CalculateAge(BirthDate As DateTime) As Integer\n    Dim HebCal As New
System.Globalization.HebrewCalendar ()\n    Dim now = DateTime.Now()\n    Dim
iAge = HebCal.GetYear(now) - HebCal.GetYear(BirthDate)\n    Dim iNowMonth =
HebCal.GetMonth(now), iBirthMonth = HebCal.GetMonth(BirthDate)\n    If iNowMonth
&lt; iBirthMonth Or (iNowMonth = iBirthMonth AndAlso HebCal.GetDayOfMonth(now)
&lt; HebCal.GetDayOfMonth(BirthDate)) Then iAge -= 1\n    Return iAge\nEnd
Function\n</code></pre>'
```

[ ]:
```
# answers after removing code
removed_code_answers[0]
```

[ ]:
```
' the following approach extract from time period library for net class datediff
considers the calendar of the culture info usage here is a solution my
suggestion that seems to have the year changing on the right date i spot tested
up to age very simple answer here is a very simple and easy to follow example
just use the current date get total hours and divide in the total hours per year
and get exactly the age months days this may work here is a datetime extender
that adds the age calculation to the datetime object this is one of the most
accurate answers that is able to resolve the birthday of th of feb compared to
any year of th feb do we need to consider people who is smaller than year as
chinese culture we describe small babies age as months or weeks below is my
implementation it is not as simple as what i imagined especially to deal with
date like this implementation has passed below test cases hope it is helpful
just because i do not think the top answer is that clear i often count on my
fingers i need to look at a calendar to work out when things change so that is
what i would do in my code running this through in linqpad gives this code in
linqpad is here this is the easiest way to answer this in a single line this
also works for leap years wow i had to give my answer here there are so many
```

answers for such a simple question i have made one small change to mark soen is answer i have rewriten the third line so that the expression can be parsed a bit more easily i have also made it into a function for the sake of clarity i have spent some time working on this and came up with this to calculate someone is age in years months and days i have tested against the feb th problem and leap years and it seems to work i would appreciate any feedback this classic question is deserving of a noda time solution usage you might also be interested in the following improvements passing in the clock as an instead of using would improve testability the target time zone will likely change so you would want a parameter as well see also my blog post on this subject handling birthdays and other anniversaries an easy to understand and simple solution however this assumes you are looking for the western idea of age and not using east asian reckoning a one liner answer to calculate how many years old a person is here is a little code sample for c# i knocked up be careful around the edge cases specifically leap years not all the above solutions take them into account pushing the answer out as a datetime can cause problems as you could end up trying to put too many days into a specific month e g days in feb this is simple and appears to be accurate for my needs i am making an assumption for the purpose of leap years that regardless of when the person chooses to celebrate the birthday they are not technically a year older until days have passed since their last birthday i e th february does not make them a year older let us know if you spot any problems check this out the simple answer to this is to apply as shown below because this is the only native method to add years to the th of feb of leap years and obtain the correct result of the th of feb for common years some feel that th of mar is the birthday of leaplings but neither net nor any official rule supports this nor does common logic explain why some born in february should have of their birthdays in another month further an age method lends itself to be added as an extension to by this you can obtain the age in the simplest possible way list item int age birthdate age now run this test the critical date example is this birth date later date age output and for the later date main problems to solve are calculate exact age in years months days etc calculate generally perceived age people usually do not care how old they exactly are they just care when their birthday in the current year is solution for is obvious solution for is the one which is not so precise in determing total age but is perceived as precise by people people also usually use it when they calculate their age manually notes to this is my preferred solution we cannot use datetime dayofyear or timespans as they shift number of days in leap years i have put there little more lines for readability just one more note i would create static overloaded methods for it one for universal usage second for usage friendliness here is a one liner it can be this simple this is a strange way to do it but if you format the date to and subtract the date of birth from the current date then drop the last digits you have got the age i do not know c# but i believe this will work in any language drop the last digits c# code or alternatively without all the type conversion in the form of an extension method error checking omitted this gives more detail to this question maybe this is what you are looking for i have a customized method to calculate age plus a bonus validation message just in case it helps method call here and pass out

datetime value mm dd yyyy if server set to usa locale replace this with anything a messagebox or any container to display remember you can format the message any way you like try this solution it is working i have created an age struct which looks like this here is a function that is serving me well no calculations very simple and here is a unit test i used scarcher is solution for an accurate year calculation of a persons age but i needed to take it further and calculate their months and days along with the years this is not a direct answer but more of a philosophical reasoning about the problem at hand from a quasi scientific point of view i would argue that the question does not specify the unit nor culture in which to measure age most answers seem to assume an integer annual representation the si unit for time is ergo the correct generic answer should be of course assuming normalized and taking no regard whatsoever to relativistic effects in the christian way of calculating age in years in finance there is a similar problem when calculating something often referred to as the day count fraction which roughly is a number of years for a given period and the age issue is really a time measuring issue example for the actual actual counting all days correctly convention another quite common way to measure time generally is by serializing the dude who named this date convention must seriously have been trippin i wonder how long we have to go before a relativistic age in seconds becomes more useful than the rough approximation of earth around sun cycles during one is lifetime so far or in other words when a period must be given a location or a function representing motion for itself to be valid how about this solution here is yet another answer this has been extensively unit tested it does look a bit quot magic quot the number is the number of days there would be in a year if every month had days the explanation of why it works lifted from here is let is set we know that what we need is either if the date has already been reached if it has not a if we have with integer division b if and we have with integer division again c if we have with integer division d if and we have with integer division again e if and we have and therefore the best way that i know of because of leap years and everything is another function not by me but found on the web and refined it a bit just two things that come into my mind what about people from countries that do not use the gregorian calendar datetime now is in the server specific culture i think i have absolutely zero knowledge about actually working with asian calendars and i do not know if there is an easy way to convert dates between calendars but just in case you are wondering about those chinese guys from the year simple code common saying from months to years old if you just for common use here is the code as your information the test result as below how come the msdn help did not tell you that it looks so obvious i have used the following for this issue i know it is not very elegant but it is working keeping it simple and possibly stupid to calculate the age with nearest age this is the version we use here it works and it is fairly simple it is the same idea as jeff is but i think it is a little clearer because it separates out the logic for subtracting one so it is a little easier to understand you could expand the ternary operator to make it even clearer if you think that sort of thing is unclear obviously this is done as an extension method on but clearly you can grab that one line of code that does the work and put it anywhere here we have another overload of the extension method that

```
passes in just for completeness with fewer conversions and utcnow this code can
take care of someone born on the feb in a leap year here is the simplest way to
calculate someone is age calculating someone is age is pretty straightforward
and here is how in order for the code to work you need a datetime object called
birthdate containing the birthday i would simply do this this way you can
calculate the exact age of a person down to the millisecond if you want using
you can apply properties whatever you want to display it i think the timespan
has all that we need in it without having to resort to or any other
approximation expanding on aug is example sql version here is a test snippet
here you have the methods i have created a sql server user defined function to
calculate someone is age given their birthdate this is useful when you need it
as part of a query i am not sure how exactly you would like it returned to you
so i just made a readable string the simplest way i have ever found is this it
works correctly for the us and western europe locales can not speak to other
locales especially places like china extra compares at most following the
initial computation of age i was looking over the answers to this and noticed
that nobody has made reference to regulatory legal implications of leap day
births for instance per wikipedia if you are born on february th in various
jurisdictions you are non leap year birthday varies in the united kingdom and
hong kong it is the ordinal day of the year so the next day march st is your
birthday in new zealand it is the previous day february th for the purposes of
driver licencing and march st for other purposes taiwan it is february th and as
near as i can tell in the us the statutes are silent on the matter leaving it up
to the common law and to how various regulatory bodies define things in their
regulations to that end an improvement it should be noted that this code assumes
a western european reckoning of age and a calendar like the gregorian calendar
that inserts a single leap day at the end of a month i use this i want to add
hebrew calendar calculations or other system globalization calendar can be used
in the same way using rewrited functions from this thread '
```

[ ]:
```
# remove stopwords from answers
preprocessed_answers = []
for i in tqdm(removed_code_answers):
    preprocessed_answers.append(remove_stopwords(i))
```

```
100%|      | 572554/572554 [1:38:33<00:00, 96.82it/s]
```

[ ]:
```
df['answers'][0]
```

[ ]:
```
'<p>The following approach (extract from <a
href="https://www.codeproject.com/Articles/168662/Time-Period-Library-for-NET"
rel="nofollow noreferrer">Time Period Library for .NET</a> class
<em>DateDiff</em>) considers the calendar of the culture
info:</p>\n\n<pre><code>//
-----------------------------------------------------------------------\nprivate
static int YearDiff( DateTime date1, DateTime date2 )\n{\n  return YearDiff(
date1, date2, DateTimeFormatInfo.CurrentInfo.Calendar );\n} // YearDiff\n\n//
```

```
----------------------------------------------------------------------\nprivate
static int YearDiff( DateTime date1, DateTime date2, Calendar calendar )\n{\n
if ( date1.Equals( date2 ) )\n  {\n    return 0;\n  }\n\n  int year1 =
calendar.GetYear( date1 );\n  int month1 = calendar.GetMonth( date1 );\n  int
year2 = calendar.GetYear( date2 );\n  int month2 = calendar.GetMonth( date2
);\n\n  // find the the day to compare\n  int compareDay = date2.Day;\n  int
compareDaysPerMonth = calendar.GetDaysInMonth( year1, month1 );\n  if (
compareDay &gt; compareDaysPerMonth )\n  {\n    compareDay =
compareDaysPerMonth;\n  }\n\n  // build the compare date\n  DateTime compareDate
= new DateTime( year1, month2, compareDay,\n    date2.Hour, date2.Minute,
date2.Second, date2.Millisecond );\n  if ( date2 &gt; date1 )\n  {\n    if (
compareDate &lt; date1 )\n    {\n      compareDate = compareDate.AddYears( 1
);\n    }\n  }\n  else\n  {\n    if ( compareDate &gt; date1 )\n    {\n
compareDate = compareDate.AddYears( -1 );\n    }\n  }\n  return year2 -
calendar.GetYear( compareDate );\n} //
YearDiff\n</code></pre>\n\n<p>Usage:</p>\n\n<pre><code>//
----------------------------------------------------------------------\npublic
void CalculateAgeSamples()\n{\n  PrintAge( new DateTime( 2000, 02, 29 ), new
DateTime( 2009, 02, 28 ) );\n  // &gt; Birthdate=29.02.2000, Age at 28.02.2009
is 8 years\n  PrintAge( new DateTime( 2000, 02, 29 ), new DateTime( 2012, 02, 28
) );\n  // &gt; Birthdate=29.02.2000, Age at 28.02.2012 is 11 years\n} //
CalculateAgeSamples\n\n//
----------------------------------------------------------------------\npublic
void PrintAge( DateTime birthDate, DateTime moment )\n{\n  Console.WriteLine(
"Birthdate={0:d}, Age at {1:d} is {2} years", birthDate, moment, YearDiff(
birthDate, moment ) );\n} // PrintAge\n</code></pre>\n<p>Here is a
solution.</p>\n\n<pre><code>DateTime dateOfBirth = new DateTime(2000, 4,
18);\nDateTime currentDate = DateTime.Now;\n\nint ageInYears = 0;\nint
ageInMonths = 0;\nint ageInDays = 0;\n\nageInDays = currentDate.Day -
dateOfBirth.Day;\nageInMonths = currentDate.Month -
dateOfBirth.Month;\nageInYears = currentDate.Year - dateOfBirth.Year;\n\nif
(ageInDays &lt; 0)\n{\n    ageInDays += DateTime.DaysInMonth(currentDate.Year,
currentDate.Month);\n    ageInMonths = ageInMonths--;\n\n    if (ageInMonths
&lt; 0)\n    {\n        ageInMonths += 12;\n        ageInYears--;\n
}\n}\n\nif (ageInMonths &lt; 0)\n{\n    ageInMonths += 12;\n
ageInYears--;\n}\n\nConsole.WriteLine("{0}, {1}, {2}", ageInYears, ageInMonths,
ageInDays);\n</code></pre>\n<p>My suggestion</p>\n\n<pre class="lang-cs
prettyprint-override"><code>int age = (int) ((DateTime.Now -
bday).TotalDays/365.242199);\n</code></pre>\n\n<p>That seems to have the year
changing on the right date. (I spot tested up to age 107.)</p>\n<p>Very simple
answer</p>\n\n<pre><code>        DateTime dob = new DateTime(1991, 3, 4); \n
DateTime now = DateTime.Now; \n        int dobDay = dob.Day, dobMonth =
dob.Month; \n        int add = -1; \n        if (dobMonth &lt; now.Month)\n
{\n            add = 0;\n        }\n        else if (dobMonth == now.Month)\n
{\n            if(dobDay &lt;= now.Day)\n            {\n                add =
0;\n            }\n            else\n            {\n                add = -1;\n
}\n        }\n        else\n        {\n            add = -1;\n        } \n
```

```
int age = now.Year - dob.Year + add;\n</code></pre>\n<p>Here is a very simple
and easy to follow example. </p>\n\n<pre><code>private int
CalculateAge()\n{\n//get birthdate\n   DateTime dtBirth =
Convert.ToDateTime(BirthDatePicker.Value);\n   int byear = dtBirth.Year;\n   int
bmonth = dtBirth.Month;\n   int bday = dtBirth.Day;\n   DateTime dtToday =
DateTime.Now;\n   int tYear = dtToday.Year;\n   int tmonth = dtToday.Month;\n
int tday = dtToday.Day;\n   int age = tYear - byear;\n   if (bmonth &lt;
tmonth)\n       age--;\n   else if (bmonth == tmonth &amp;&amp; bday&gt;tday)\n
{\n       age--;\n   }\nreturn age;\n}\n</code></pre>\n<p>Just use:</p>\n\n<pre
class="lang-cs prettyprint-override"><code>(DateTime.Now - myDate).TotalHours /
8766.0\n</code></pre>\n\n<p>The current date - <code>myDate = TimeSpan</code>,
get total hours and divide in the total hours per year and get exactly the
age/months/days…</p>\n<p>This may work:</p>\n\n<pre class="lang-cs
prettyprint-override"><code>public override bool IsValid(DateTime value)\n{\n
_dateOfBirth =  value;\n    var yearsOld = (double)
(DateTime.Now.Subtract(_dateOfBirth).TotalDays/365);\n    if (yearsOld &gt;
18)\n        return true;\n    return false;\n}\n</code></pre>\n<p>Here\'s a
DateTime extender that adds the age calculation to the DateTime
object.</p>\n\n<pre class="lang-cs prettyprint-override"><code>public static
class AgeExtender\n{\n    public static int GetAge(this DateTime dt)\n    {\n
int d = int.Parse(dt.ToString("yyyyMMdd"));\n        int t =
int.Parse(DateTime.Today.ToString("yyyyMMdd"));\n        return (t-d)/10000;\n
}\n}\n</code></pre>\n<p>This is one of the most accurate answers that is able to
resolve the birthday of 29th of Feb compared to any year of 28th
Feb.</p>\n\n<pre class="lang-cs prettyprint-override"><code>public int
GetAge(DateTime birthDate)\n{\n    int age = DateTime.Now.Year -
birthDate.Year;\n\n    if (birthDate.DayOfYear &gt; DateTime.Now.DayOfYear)\n
age--;\n\n    return age;\n}\n\n\n\n\n</code></pre>\n<p>Do we need to consider
people who is smaller than 1 year? as Chinese culture, we describe small
babies\' age as 2 months or 4 weeks. </p>\n\n<p>Below is my implementation, it
is not as simple as what I imagined, especially to deal with date like 2/28.
</p>\n\n<pre><code>public static string HowOld(DateTime birthday, DateTime
now)\n{\n    if (now &lt; birthday)\n        throw new
ArgumentOutOfRangeException("birthday must be less than now.");\n\n    TimeSpan
diff = now - birthday;\n    int diffDays = (int)diff.TotalDays;\n\n    if
(diffDays &gt; 7)//year, month and week\n    {\n        int age = now.Year -
birthday.Year;\n\n        if (birthday &gt; now.AddYears(-age))\n
age--;\n\n        if (age &gt; 0)\n        {\n            return age + (age &gt;
1 ? " years" : " year");\n        }\n        else\n        {// month and week\n
DateTime d = birthday;\n            int diffMonth = 1;\n\n            while
(d.AddMonths(diffMonth) &lt;= now)\n            {\n
diffMonth++;\n            }\n\n            age = diffMonth-1;\n\n            if
(age == 1 &amp;&amp; d.Day &gt; now.Day)\n                age--;\n\n
if (age &gt; 0)\n            {\n                return age + (age &gt; 1 ? "
months" : " month");\n            }\n            else\n            {\n
age = diffDays / 7;\n                return age + (age &gt; 1 ? " weeks" : "
week");\n            }\n        }\n    }\n    else if (diffDays &gt; 0)\n    {\n
```

int age = diffDays;\n   return age + (age > 1 ? " days" : " day");\n   }\n   else\n   {\n   int age = diffDays;\n   return "just born";\n   }\n}\n</code></pre>\n\n This implementation has passed below test cases. \n\n<pre><code>[TestMethod]\npublic void TestAge()\n{\n   string age = HowOld(new DateTime(2011, 1, 1), new DateTime(2012, 11, 30));\n   Assert.AreEqual("1 year", age);\n\n   age = HowOld(new DateTime(2011, 11, 30), new DateTime(2012, 11, 30));\n   Assert.AreEqual("1 year", age);\n\n   age = HowOld(new DateTime(2001, 1, 1), new DateTime(2012, 11, 30));\n   Assert.AreEqual("11 years", age);\n\n   age = HowOld(new DateTime(2012, 1, 1), new DateTime(2012, 11, 30));\n   Assert.AreEqual("10 months", age);\n\n   age = HowOld(new DateTime(2011, 12, 1), new DateTime(2012, 11, 30));\n   Assert.AreEqual("11 months", age);\n\n   age = HowOld(new DateTime(2012, 10, 1), new DateTime(2012, 11, 30));\n   Assert.AreEqual("1 month", age);\n\n   age = HowOld(new DateTime(2008, 2, 28), new DateTime(2009, 2, 28));\n   Assert.AreEqual("1 year", age);\n\n   age = HowOld(new DateTime(2008, 3, 28), new DateTime(2009, 2, 28));\n   Assert.AreEqual("11 months", age);\n\n   age = HowOld(new DateTime(2008, 3, 28), new DateTime(2009, 3, 28));\n   Assert.AreEqual("1 year", age);\n\n   age = HowOld(new DateTime(2009, 1, 28), new DateTime(2009, 2, 28));\n   Assert.AreEqual("1 month", age);\n\n   age = HowOld(new DateTime(2009, 2, 1), new DateTime(2009, 3, 1));\n   Assert.AreEqual("1 month", age);\n\n   // NOTE.\n   // new DateTime(2008, 1, 31).AddMonths(1) == new DateTime(2009, 2, 28);\n   // new DateTime(2008, 1, 28).AddMonths(1) == new DateTime(2009, 2, 28);\n   age = HowOld(new DateTime(2009, 1, 31), new DateTime(2009, 2, 28));\n   Assert.AreEqual("4 weeks", age);\n\n   age = HowOld(new DateTime(2009, 2, 1), new DateTime(2009, 2, 28));\n   Assert.AreEqual("3 weeks", age);\n\n   age = HowOld(new DateTime(2009, 2, 1), new DateTime(2009, 3, 1));\n   Assert.AreEqual("1 month", age);\n\n   age = HowOld(new DateTime(2012, 11, 5), new DateTime(2012, 11, 30));\n   Assert.AreEqual("3 weeks", age);\n\n   age = HowOld(new DateTime(2012, 11, 1), new DateTime(2012, 11, 30));\n   Assert.AreEqual("4 weeks", age);\n\n   age = HowOld(new DateTime(2012, 11, 20), new DateTime(2012, 11, 30));\n   Assert.AreEqual("1 week", age);\n\n   age = HowOld(new DateTime(2012, 11, 25), new DateTime(2012, 11, 30));\n   Assert.AreEqual("5 days", age);\n\n   age = HowOld(new DateTime(2012, 11, 29), new DateTime(2012, 11, 30));\n   Assert.AreEqual("1 day", age);\n\n   age = HowOld(new DateTime(2012, 11, 30), new DateTime(2012, 11, 30));\n   Assert.AreEqual("just born", age);\n\n   age = HowOld(new DateTime(2000, 2, 29), new DateTime(2009, 2, 28));\n   Assert.AreEqual("8 years", age);\n\n   age = HowOld(new DateTime(2000, 2, 29), new DateTime(2009, 3, 1));\n   Assert.AreEqual("9 years", age);\n\n   Exception e = null;\n\n   try\n   {\n   age = HowOld(new DateTime(2012, 12, 1), new DateTime(2012, 11, 30));\n   }\n   catch (ArgumentOutOfRangeException ex)\n   {\n   e = ex;\n   }\n\n   Assert.IsTrue(e != null);\n}\n</code></pre>\n\n Hope it\'s helpful. \n Just because I don\'t think the top answer is that clear: \n\n<pre class="lang-cs prettyprint-override"><code>public static int GetAgeByLoop(DateTime birthday)\n{\n   var age = -1;\n\n   for (var date = birthday; date < DateTime.Today; date = date.AddYears(1))\n   {\n

age++;\n   }\n\n   return age;\n}\n</code></pre>\n I often count on my fingers. I need to look at a calendar to work out when things change. So that\'s what I\'d do in my code: \n\n<pre><code>int AgeNow(DateTime birthday)\n{\n return AgeAt(DateTime.Now, birthday);\n}\n\nint AgeAt(DateTime now, DateTime birthday)\n{\n   return AgeAt(now, birthday, CultureInfo.CurrentCulture.Calendar);\n}\n\nint AgeAt(DateTime now, DateTime birthday, Calendar calendar)\n{\n   // My age has increased on the morning of my\n   // birthday even though I was born in the evening.\n   now = now.Date;\n   birthday = birthday.Date;\n\n   var age = 0;\n   if (now <= birthday) return age; // I am zero now if I am to be born tomorrow.\n\n   while (calendar.AddYears(birthday, age + 1) <= now)\n   {\n   age++;\n   }\n return age;\n}\n</code></pre>\n\n Running this through in LINQPad gives this: \n\n<pre><code>PASSED: someone born on 28 February 1964 is age 4 on 28 February 1968\nPASSED: someone born on 29 February 1964 is age 3 on 28 February 1968\nPASSED: someone born on 31 December 2016 is age 0 on 01 January 2017\n</code></pre>\n\n Code in LINQPad is here. \n This is the easiest way to answer this in a single line. \n\n<pre class="lang-cs prettyprint-override"><code>DateTime Dob = DateTime.Parse("1985-04-24");\n\nint Age = DateTime.MinValue.AddDays(DateTime.Now.Subtract(Dob).TotalHours/24).Year - 1;\n</code></pre>\n\n This also works for leap years. \n<pre class="lang-cs prettyprint-override"><code>private int GetYearDiff(DateTime start, DateTime end)\n{\n   int diff = end.Year - start.Year;\n   if (end.DayOfYear < start.DayOfYear) { diff -= 1; }\n   return diff;\n}\n[Fact]\npublic void GetYearDiff_WhenCalls_ShouldReturnCorrectYearDiff()\n{\n   //arrange\n   var now = DateTime.Now;\n   //act\n   //assert\n   Assert.Equal(24, GetYearDiff(new DateTime(1992, 7, 9), now)); // passed\n   Assert.Equal(24, GetYearDiff(new DateTime(1992, now.Month, now.Day), now)); // passed\n   Assert.Equal(23, GetYearDiff(new DateTime(1992, 12, 9), now)); // passed\n}\n</code></pre>\n Wow, I had to give my answer here… There are so many answers for such a simple question. \n\n<pre><code>private int CalcularIdade(DateTime dtNascimento)\n   {\n   var nHoje = Convert.ToInt32(DateTime.Today.ToString("yyyyMMdd"));\n   var nAniversario = Convert.ToInt32(dtNascimento.ToString("yyyyMMdd"));\n\n   double diff = (nHoje - nAniversario) / 10000;\n\n   var ret = Convert.ToInt32(Math.Truncate(diff));\n\n   return ret;\n   }\n</code></pre>\n I\'ve made one small change to Mark Soen\'s answer: I\'ve rewriten the third line so that the expression can be parsed a bit more easily. \n\n<pre><code>public int AgeInYears(DateTime bday)\n{\n   DateTime now = DateTime.Today;\n   int age = now.Year - bday.Year;   \n   if (bday.AddYears(age) > now) \n age--;\n   return age;\n}\n</code></pre>\n\n I\'ve also made it into a function for the sake of clarity. \n I\'ve spent some time working on this and came up with this to calculate someone\'s age in years, months and days.

I\'ve tested against the Feb 29th problem and leap years and it seems to work, I\'d appreciate any feedback: \n\n<pre><code>public void LoopAge(DateTime myDOB, DateTime FutureDate)\n{\n   int years = 0;\n   int months = 0;\n   int days = 0;\n\n   DateTime tmpMyDOB = new DateTime(myDOB.Year, myDOB.Month, 1);\n\n   DateTime tmpFutureDate = new DateTime(FutureDate.Year, FutureDate.Month, 1);\n\n   while (tmpMyDOB.AddYears(years).AddMonths(months) < tmpFutureDate)\n   {\n   months++;\n\n   if (months > 12)\n   {\n   years++;\n   months = months - 12;\n   }\n   }\n\n   if (FutureDate.Day >= myDOB.Day)\n   {\n   days = days + FutureDate.Day - myDOB.Day;\n   }\n   else\n   {\n   months--;\n\n   if (months < 0)\n   {\n   years--;\n   months = months + 12;\n   }\n\n   days +=\n   DateTime.DaysInMonth(\n   FutureDate.AddMonths(-1).Year, FutureDate.AddMonths(-1).Month\n   ) + FutureDate.Day - myDOB.Day;\n\n   }\n\n   //add an extra day if the dob is a leap day\n   if (DateTime.IsLeapYear(myDOB.Year) && myDOB.Month == 2 && myDOB.Day == 29)\n   {\n   //but only if the future date is less than 1st March\n   if (FutureDate >= new DateTime(FutureDate.Year, 3, 1))\n   days++;\n   }\n\n}\n</code></pre>\n This classic question is deserving of a Noda Time solution. \n\n<pre><code>static int GetAge(LocalDate dateOfBirth)\n{\n   Instant now = SystemClock.Instance.Now;\n\n   // The target time zone is important.\n   // It should align with the *current physical location* of the person\n   // you are talking about. When the whereabouts of that person are unknown,\n   // then you use the time zone of the person who is *asking* for the age.\n   // The time zone of birth is irrelevant!\n\n   DateTimeZone zone = DateTimeZoneProviders.Tzdb["America/New_York"];\n\n   LocalDate today = now.InZone(zone).Date;\n\n   Period period = Period.Between(dateOfBirth, today, PeriodUnits.Years);\n\n   return (int) period.Years;\n}\n</code></pre>\n\n Usage: \n\n<pre><code>LocalDate dateOfBirth = new LocalDate(1976, 8, 27);\nint age = GetAge(dateOfBirth);\n</code></pre>\n\n You might also be interested in the following improvements: \n\n
\n Passing in the clock as an <code>IClock</code>, instead of using <code>SystemClock.Instance</code>, would improve testability. \n The target time zone will likely change, so you\'d want a <code>DateTimeZone</code> parameter as well. \n
\n\n See also my blog post on this subject: Handling Birthdays, and Other Anniversaries \n An easy to understand and simple solution. \n\n<pre class="lang-cs prettyprint-override"><code>\n// Save today\'s date.\nvar today = DateTime.Today;\n\n// Calculate the age.\nvar age = today.Year - birthdate.Year;\n\n// Go back to the year the person was born in case of a leap year\nif (birthdate.Date > today.AddYears(-age))
age--;\n</code></pre>\n\n However, this assumes you are looking for the western idea of age and not using East Asian reckoning. \n A one-liner

answer: \n\n<pre class="lang-cs prettyprint-override"><code>DateTime dateOfBirth = Convert.ToDateTime("01/16/1990");\nvar age = ((DateTime.Now - dateOfBirth).Days) / 365;\n</code></pre>\n To calculate how many years old a person is, \n\n<pre class="lang-cs prettyprint-override"><code>DateTime dateOfBirth;\n\nint ageInYears = DateTime.Now.Year - dateOfBirth.Year;\n\nif (dateOfBirth > today.AddYears(-ageInYears)) ageInYears --;\n</code></pre>\n Here\'s a little code sample for C# I knocked up, be careful around the edge cases specifically leap years, not all the above solutions take them into account. Pushing the answer out as a DateTime can cause problems as you could end up trying to put too many days into a specific month e.g. 30 days in Feb. \n\n<pre><code>public string LoopAge(DateTime myDOB, DateTime FutureDate)\n{\n   int years = 0;\n   int months = 0;\n   int days = 0;\n\n   DateTime tmpMyDOB = new DateTime(myDOB.Year, myDOB.Month, 1);\n\n DateTime tmpFutureDate = new DateTime(FutureDate.Year, FutureDate.Month, 1);\n\n while (tmpMyDOB.AddYears(years).AddMonths(months) < tmpFutureDate)\n   {\n months++;\n   if (months > 12)\n   {\n   years++;\n months = months - 12;\n   }\n   }\n\n   if (FutureDate.Day >= myDOB.Day)\n   {\n   days = days + FutureDate.Day - myDOB.Day;\n   }\n else\n   {\n   months--;\n   if (months < 0)\n   {\n years--;\n   months = months + 12;\n   }\n   days = days + (DateTime.DaysInMonth(FutureDate.AddMonths(-1).Year, FutureDate.AddMonths(-1).Month) + FutureDate.Day) - myDOB.Day;\n\n   }\n\n //add an extra day if the dob is a leap day\n   if (DateTime.IsLeapYear(myDOB.Year) && myDOB.Month == 2 && myDOB.Day == 29)\n   {\n   //but only if the future date is less than 1st March\n   if(FutureDate >= new DateTime(FutureDate.Year, 3,1))\n days++;\n   }\n\n   return "Years: " + years + " Months: " + months + " Days: " + days;\n}\n</code></pre>\n This is simple and appears to be accurate for my needs. I am making an assumption for the purpose of leap years that regardless of when the person chooses to celebrate the birthday they are not technically a year older until 365 days have passed since their last birthday (i.e 28th February does not make them a year older). \n<pre><code>DateTime now = DateTime.Today;\nDateTime birthday = new DateTime(1991, 02, 03);//3rd feb\n\nint age = now.Year - birthday.Year;\n\nif (now.Month < birthday.Month || (now.Month == birthday.Month && now.Day < birthday.Day))//not had bday this year yet\n age--;\n\nreturn age;\n</code></pre>\n Let us know if you spot any problems ;) \n Check this out: \n\n<pre class="lang-cs prettyprint-override"><code>TimeSpan ts = DateTime.Now.Subtract(Birthdate);\nage = (byte)(ts.TotalDays / 365.25);\n</code></pre>\n The simple answer to this is to apply <code>AddYears</code> as shown below because this is the only native method to add years to the 29th of Feb. of leap years and obtain the correct result of the 28th of Feb. for common years. \n\n Some feel that 1th of Mar. is the birthday of leaplings but neither .Net nor any official rule supports this, nor does common logic explain why some born in February should have 75% of their birthdays in another month. \n\n Further, an Age method lends itself to be added as an extension to <code>DateTime</code>. By this you can obtain the age in the simplest possible way: \n\n<ol>\n<li>List

item
\n
\n\n<p>int age =
birthDate.Age();</p>\n\n<pre><code>public static class
DateTimeExtensions\n{\n   /// <summary>\n   /// Calculates the age in
years of the current System.DateTime object today.\n   /// </summary>\n
/// <param name="birthDate">The date of birth</param>\n   ///
<returns>Age in years today. 0 is returned for a future date of
birth.</returns>\n   public static int Age(this DateTime birthDate)\n
{\n   return Age(birthDate, DateTime.Today);\n   }\n\n   ///
<summary>\n   /// Calculates the age in years of the current
System.DateTime object on a later date.\n   /// </summary>\n   ///
<param name="birthDate">The date of birth</param>\n   /// <param
name="laterDate">The date on which to calculate the age.</param>\n
/// <returns>Age in years on a later day. 0 is returned as
minimum.</returns>\n   public static int Age(this DateTime birthDate,
DateTime laterDate)\n   {\n   int age;\n   age = laterDate.Year -
birthDate.Year;\n\n   if (age > 0)\n   {\n   age -=
Convert.ToInt32(laterDate.Date < birthDate.Date.AddYears(age));\n   }\n
else\n   {\n   age = 0;\n   }\n\n   return age;\n
}\n}\n</code></pre>\n\n<p>Now, run this test:</p>\n\n<pre><code>class
Program\n{\n   static void Main(string[] args)\n   {\n   RunTest();\n
}\n\n   private static void RunTest()\n   {\n   DateTime birthDate = new
DateTime(2000, 2, 28);\n   DateTime laterDate = new DateTime(2011, 2,
27);\n   string iso = "yyyy-MM-dd";\n\n   for (int i = 0; i < 3;
i++)\n   {\n   for (int j = 0; j < 3; j++)\n   {\n
Console.WriteLine("Birth date: " + birthDate.AddDays(i).ToString(iso) + " Later
date: " + laterDate.AddDays(j).ToString(iso) + " Age: " +
birthDate.AddDays(i).Age(laterDate.AddDays(j)).ToString());\n   }\n
}\n\n   Console.ReadKey();\n   }\n}\n</code></pre>\n\n<p>The critical date
example is this:</p>\n\n<p>Birth date: 2000-02-29 Later date:
2011-02-28 Age: 11</p>\n\n<p>Output:</p>\n\n<pre><code>{\n   Birth
date: 2000-02-28 Later date: 2011-02-27 Age: 10\n   Birth date: 2000-02-28
Later date: 2011-02-28 Age: 11\n   Birth date: 2000-02-28 Later date:
2011-03-01 Age: 11\n   Birth date: 2000-02-29 Later date: 2011-02-27 Age:
10\n   Birth date: 2000-02-29 Later date: 2011-02-28 Age: 11\n   Birth date:
2000-02-29 Later date: 2011-03-01 Age: 11\n   Birth date: 2000-03-01 Later
date: 2011-02-27 Age: 10\n   Birth date: 2000-03-01 Later date: 2011-02-28
Age: 10\n   Birth date: 2000-03-01 Later date: 2011-03-01 Age:
11\n}\n</code></pre>\n\n<p>And for the later date
2012-02-28:</p>\n\n<pre><code>{\n   Birth date: 2000-02-28 Later date:
2012-02-28 Age: 12\n   Birth date: 2000-02-28 Later date: 2012-02-29 Age:
12\n   Birth date: 2000-02-28 Later date: 2012-03-01 Age: 12\n   Birth date:
2000-02-29 Later date: 2012-02-28 Age: 11\n   Birth date: 2000-02-29 Later
date: 2012-02-29 Age: 12\n   Birth date: 2000-02-29 Later date: 2012-03-01
Age: 12\n   Birth date: 2000-03-01 Later date: 2012-02-28 Age: 11\n   Birth
date: 2000-03-01 Later date: 2012-02-29 Age: 11\n   Birth date: 2000-03-01
Later date: 2012-03-01 Age: 12\n}\n</code></pre>\n<p>2 Main problems to solve
are:</p>\n\n<p>1. Calculate Exact age - in years, months, days,

etc. \n\n 2. Calculate Generally perceived age - people usually do not care how old they exactly are, they just care when their birthday in the current year is. \n\n<hr>\n\n Solution for 1 is obvious: \n\n<pre><code>DateTime birth = DateTime.Parse("1.1.2000");\nDateTime today = DateTime.Today;   //we usually don\'t care about birth time\nTimeSpan age = today - birth;   //.NET FCL should guarantee this as precise\ndouble ageInDays = age.TotalDays;   //total number of days … also precise\ndouble daysInYear = 365.2425;   //statistical value for 400 years\ndouble ageInYears = ageInDays / daysInYear;   //can be shifted … not so precise\n</code></pre>\n\n<hr>\n\n Solution for 2 is the one which is not so precise in determing total age, but is perceived as precise by people. People also usually use it, when they calculate their age "manually": \n\n<pre><code>DateTime birth = DateTime.Parse("1.1.2000");\nDateTime today = DateTime.Today;\nint age = today.Year - birth.Year;   //people perceive their age in years\n\nif (today.Month < birth.Month ||\n   ((today.Month == birth.Month) && (today.Day < birth.Day)))\n{\n age--; //birthday in current year not yet reached, we are 1 year younger ;)\n   //+ no birthday for 29.2. guys … sorry, just wrong date for birth\n}\n</code></pre>\n\n Notes to 2.: \n\n \n This is my preferred solution \n We cannot use DateTime.DayOfYear or TimeSpans, as they shift number of days in leap years \n I have put there little more lines for readability \n \n\n Just one more note … I would create 2 static overloaded methods for it, one for universal usage, second for usage-friendliness: \n\n<pre><code>public static int GetAge(DateTime bithDay, DateTime today) \n{ \n //chosen solution method body\n}\n\npublic static int GetAge(DateTime birthDay) \n{ \n return GetAge(birthDay, DateTime.Now);\n}\n</code></pre>\n Here\'s a one-liner: \n\n<pre class="lang-cs prettyprint-override"><code>int age = new DateTime(DateTime.Now.Subtract(birthday).Ticks).Year-1;\n</code></pre>\n It can be this simple: \n\n<pre class="lang-cs prettyprint-override"><code>int age = DateTime.Now.AddTicks(0 - dob.Ticks).Year - 1;\n</code></pre>\n This is a strange way to do it, but if you format the date to <code>yyyymmdd</code> and subtract the date of birth from the current date then drop the last 4 digits you\'ve got the age :) \n\n I don\'t know C#, but I believe this will work in any language. \n\n<pre class="lang-cs prettyprint-override"><code>20080814 - 19800703 = 280111 \n</code></pre>\n\n Drop the last 4 digits = <code>28</code>. \n\n C# Code: \n\n<pre class="lang-cs prettyprint-override"><code>int now = int.Parse(DateTime.Now.ToString("yyyyMMdd"));\nint dob = int.Parse(dateOfBirth.ToString("yyyyMMdd"));\nint age = (now - dob) / 10000;\n</code></pre>\n\n Or alternatively without all the type conversion in the form of an extension method. Error checking omitted: \n\n<pre class="lang-cs prettyprint-override"><code>public static Int32 GetAge(this DateTime dateOfBirth)\n{\n   var today = DateTime.Today;\n\n   var a = (today.Year * 100 + today.Month) * 100 + today.Day;\n   var b = (dateOfBirth.Year * 100 + dateOfBirth.Month) * 100 + dateOfBirth.Day;\n\n   return (a - b) / 10000;\n}\n</code></pre>\n This gives "more detail" to this

question. Maybe this is what you\'re looking for \n\n<pre><code>DateTime birth = new DateTime(1974, 8, 29);\nDateTime today = DateTime.Now;\nTimeSpan span = today - birth;\nDateTime age = DateTime.MinValue + span;\n\n// Make adjustment due to MinValue equalling 1/1/1\nint years = age.Year - 1;\nint months = age.Month - 1;\nint days = age.Day - 1;\n\n// Print out not only how many years old they are but give months and days as well\nConsole.Write("{0} years, {1} months, {2} days", years, months, days);\n</code></pre>\n I have a customized method to calculate age, plus a bonus validation message just in case it helps: \n\n<pre><code>public void GetAge(DateTime dob, DateTime now, out int years, out int months, out int days)\n{\n   years = 0;\n   months = 0;\n   days = 0;\n\n   DateTime tmpdob = new DateTime(dob.Year, dob.Month, 1);\n   DateTime tmpnow = new DateTime(now.Year, now.Month, 1);\n\n   while (tmpdob.AddYears(years).AddMonths(months) < tmpnow)\n   {\n   months++;\n   if (months > 12)\n   {\n   years++;\n   months = months - 12;\n   }\n   }\n\n   if (now.Day >= dob.Day)\n   days = days + now.Day - dob.Day;\n   else\n   {\n   months--;\n   if (months < 0)\n   {\n   years--;\n   months = months + 12;\n   }\n   days += DateTime.DaysInMonth(now.AddMonths(-1).Year, now.AddMonths(-1).Month) + now.Day - dob.Day;\n   }\n\n   if (DateTime.IsLeapYear(dob.Year) && dob.Month == 2 && dob.Day == 29 && now >= new DateTime(now.Year, 3, 1))\n   days++;\n\n}\n\nprivate string ValidateDate(DateTime dob) //This method will validate the date\n{\n   int Years = 0; int Months = 0; int Days = 0;\n\n   GetAge(dob, DateTime.Now, out Years, out Months, out Days);\n\n   if (Years < 18)\n   message = Years + " is too young. Please try again on your 18th birthday.";\n   else if (Years >= 65)\n   message = Years + " is too old. Date of Birth must not be 65 or older.";\n   else\n   return null; //Denotes validation passed\n}\n</code></pre>\n\n Method call here and pass out datetime value (MM/dd/yyyy if server set to USA locale). Replace this with anything a messagebox or any container to display: \n\n<pre><code>DateTime dob = DateTime.Parse("03/10/1982"); \n\nstring message = ValidateDate(dob);\n\nlbldatemessage.Visible = !StringIsNullOrWhitespace(message);\nlbldatemessage.Text = message ?? ""; //Ternary if message is null then default to empty string\n</code></pre>\n\n Remember you can format the message any way you like. \n Try this solution, it\'s working. \n\n<pre class="lang-cs prettyprint-override"><code>int age = (Int32.Parse(DateTime.Today.ToString("yyyyMMdd")) - \n Int32.Parse(birthday.ToString("yyyyMMdd rawrrr"))) / 10000;\n</code></pre>\n I\'ve created an Age struct, which looks like this: \n\n<pre><code>public struct Age : IEquatable<Age>, IComparable<Age>\n{\n   private readonly int _years;\n   private readonly int _months;\n   private readonly int _days;\n\n   public int Years { get { return _years; } }\n   public int Months { get { return _months; } }\n   public int Days { get { return _days; } }\n\n   public Age(int years, int months, int days) : this()\n   {\n   _years = years;\n   _months = months;\n   _days = days;\n   }\n\n   public static Age CalculateAge(

```
DateTime dateOfBirth, DateTime date )\n    {\n        // Here is some logic that
ressembles Mike\'s solution, although it\n        // also takes into account
months &amp; days.\n        // Ommitted for brevity.\n        return new Age
(years, months, days);\n    }\n\n    // Ommited Equality, Comparable,
GetHashCode, functionality for brevity.\n}\n</code></pre>\n<p>Here is a function
that is serving me well. No calculations , very simple.</p>\n<pre><code>
public static string ToAge(this DateTime dob, DateTime? toDate = null)\n    {\n
if (!toDate.HasValue)\n            toDate = DateTime.Now;\n        var now =
toDate.Value;\n\n        if (now.CompareTo(dob) &lt; 0)\n            return
&quot;Future date&quot;;\n\n        int years = now.Year - dob.Year;\n
int months = now.Month - dob.Month;\n        int days = now.Day - dob.Day;\n\n
if (days &lt; 0)\n        {\n            months--;\n            days =
DateTime.DaysInMonth(dob.Year, dob.Month) - dob.Day + now.Day;\n        }\n\n
if (months &lt; 0)\n        {\n            years--;\n            months = 12 +
months;\n        }\n\n\n        return string.Format(&quot;{0} year(s), {1}
month(s), {2} days(s)&quot;,\n            years,\n            months,\n
days);\n    }\n</code></pre>\n<p>And here is a unit test:</p>\n<pre><code>
[Test]\n    public void ToAgeTests()\n    {\n        var date = new
DateTime(2000, 1, 1);\n        Assert.AreEqual(&quot;0 year(s), 0 month(s), 1
days(s)&quot;, new DateTime(1999, 12, 31).ToAge(date));\n
Assert.AreEqual(&quot;0 year(s), 0 month(s), 0 days(s)&quot;, new DateTime(2000,
1, 1).ToAge(date));\n        Assert.AreEqual(&quot;1 year(s), 0 month(s), 0
days(s)&quot;, new DateTime(1999, 1, 1).ToAge(date));\n
Assert.AreEqual(&quot;0 year(s), 11 month(s), 0 days(s)&quot;, new
DateTime(1999, 2, 1).ToAge(date));\n        Assert.AreEqual(&quot;0 year(s), 10
month(s), 25 days(s)&quot;, new DateTime(1999, 2, 4).ToAge(date));\n
Assert.AreEqual(&quot;0 year(s), 10 month(s), 1 days(s)&quot;, new
DateTime(1999, 2, 28).ToAge(date));\n\n        date = new DateTime(2000, 2,
15);\n        Assert.AreEqual(&quot;0 year(s), 0 month(s), 28 days(s)&quot;, new
DateTime(2000, 1, 18).ToAge(date));\n    }\n</code></pre>\n<p>I used
ScArcher2\'s solution for an accurate Year calculation of a persons age but I
needed to take it further and calculate their Months and Days along with the
Years.</p>\n\n<pre><code>    public static Dictionary&lt;string,int&gt;
CurrentAgeInYearsMonthsDays(DateTime? ndtBirthDate, DateTime? ndtReferralDate)\n
{\n
//---------------------------------------------------------------------\n
// Can\'t determine age if we don\'t have a dates.\n
//---------------------------------------------------------------------\n
if (ndtBirthDate == null) return null;\n        if (ndtReferralDate == null)
return null;\n\n        DateTime dtBirthDate =
Convert.ToDateTime(ndtBirthDate);\n        DateTime dtReferralDate =
Convert.ToDateTime(ndtReferralDate);\n\n
//---------------------------------------------------------------------\n
// Create our Variables\n
//---------------------------------------------------------------------\n
Dictionary&lt;string, int&gt; dYMD = new Dictionary&lt;string,int&gt;();\n
int iNowDate, iBirthDate, iYears, iMonths, iDays;\n        string sDif = "";\n\n
```

```
//-------------------------------------------------------------------------\n
// Store off current date/time and DOB into local variables\n
//-------------------------------------------------------------------------- \n
iNowDate = int.Parse(dtReferralDate.ToString("yyyyMMdd"));\n        iBirthDate =
int.Parse(dtBirthDate.ToString("yyyyMMdd"));\n\n
//-------------------------------------------------------------------------\n
// Calculate Years\n
//-------------------------------------------------------------------------\n
sDif = (iNowDate - iBirthDate).ToString();\n        iYears =
int.Parse(sDif.Substring(0, sDif.Length - 4));\n\n
//-------------------------------------------------------------------------\n
// Store Years in Return Value\n
//-------------------------------------------------------------------------\n
dYMD.Add("Years", iYears);\n\n
//-------------------------------------------------------------------------\n
// Calculate Months\n
//-------------------------------------------------------------------------\n
if (dtBirthDate.Month &gt; dtReferralDate.Month)\n            iMonths = 12 -
dtBirthDate.Month + dtReferralDate.Month - 1;\n        else\n            iMonths
= dtBirthDate.Month - dtReferralDate.Month;\n\n
//-------------------------------------------------------------------------\n
// Store Months in Return Value\n
//-------------------------------------------------------------------------\n
dYMD.Add("Months", iMonths);\n\n
//-------------------------------------------------------------------------\n
// Calculate Remaining Days\n
//-------------------------------------------------------------------------\n
if (dtBirthDate.Day &gt; dtReferralDate.Day)\n            //Logic: Figure out
the days in month previous to the current month, or the admitted month.\n
//       Subtract the birthday from the total days which will give us how many
days the person has lived since their birthdate day the previous month.\n
//       then take the referral date and simply add the number of days the
person has lived this month.\n\n            //If referral date is january, we
need to go back to the following year\'s December to get the days in that
month.\n            if (dtReferralDate.Month == 1)\n                iDays =
DateTime.DaysInMonth(dtReferralDate.Year - 1, 12) - dtBirthDate.Day +
dtReferralDate.Day;       \n            else\n                iDays =
DateTime.DaysInMonth(dtReferralDate.Year, dtReferralDate.Month - 1) -
dtBirthDate.Day + dtReferralDate.Day;       \n        else\n            iDays =
dtReferralDate.Day - dtBirthDate.Day;             \n\n
//-------------------------------------------------------------------------\n
// Store Days in Return Value\n
//-------------------------------------------------------------------------\n
dYMD.Add("Days", iDays);\n\n        return dYMD;\n}\n</code></pre>\n<p>This is
not a direct answer, but more of a philosophical reasoning about the problem at
hand from a quasi-scientific point of view.</p>\n\n<p>I would argue that the
question does not specify the unit nor culture in which to measure age, most
```

answers seem to assume an integer annual representation. The SI-unit for time is <code>second</code>, ergo the correct generic answer should be (of course assuming normalized <code>DateTime</code> and taking no regard whatsoever to relativistic effects): \n\n<pre><code>var lifeInSeconds = (DateTime.Now.Ticks - then.Ticks)/TickFactor;\n</code></pre>\n\n In the Christian way of calculating age in years: \n\n<pre><code>var then = … // Then, in this case the birthday\nvar now = DateTime.UtcNow;\nint age = now.Year - then.Year;\nif (now.AddYears(-age) < then) age--;\n</code></pre>\n\n In finance there is a similar problem when calculating something often referred to as the Day Count Fraction, which roughly is a number of years for a given period. And the age issue is really a time measuring issue. \n\n Example for the actual/actual (counting all days "correctly") convention: \n\n<pre><code>DateTime start, end = … // Whatever, assume start is before end\n\ndouble startYearContribution = 1 - (double) start.DayOfYear / (double) (DateTime.IsLeapYear(start.Year) ? 366 : 365);\ndouble endYearContribution = (double)end.DayOfYear / (double)(DateTime.IsLeapYear(end.Year) ? 366 : 365);\ndouble middleContribution = (double) (end.Year - start.Year - 1);\n\ndouble DCF = startYearContribution + endYearContribution + middleContribution;\n</code></pre>\n\n Another quite common way to measure time generally is by "serializing" (the dude who named this date convention must seriously have been trippin\'): \n\n<pre><code>DateTime start, end = … // Whatever, assume start is before end\nint days = (end - start).Days;\n</code></pre>\n\n I wonder how long we have to go before a relativistic age in seconds becomes more useful than the rough approximation of earth-around-sun-cycles during one\'s lifetime so far :) Or in other words, when a period must be given a location or a function representing motion for itself to be valid :) \n How about this solution? \n\n<pre><code>static string CalcAge(DateTime birthDay)\n{\n DateTime currentDate = DateTime.Now;   \n   int approximateAge = currentDate.Year - birthDay.Year;\n   int daysToNextBirthDay = (birthDay.Month * 30 + birthDay.Day) - \n   (currentDate.Month * 30 + currentDate.Day) ;\n\n   if (approximateAge == 0 || approximateAge == 1)\n   {\n   int month = Math.Abs(daysToNextBirthDay / 30);\n   int days = Math.Abs(daysToNextBirthDay % 30);\n\n   if (month == 0)\n return "Your age is: " + daysToNextBirthDay + " days";\n\n   return "Your age is: " + month + " months and " + days + " days"; ;\n   }\n\n   if (daysToNextBirthDay > 0)\n   return "Your age is: " + --approximateAge + " Years";\n\n   return "Your age is: " + approximateAge + " Years"; ;\n}\n</code></pre>\n Here\'s yet another answer: \n<pre><code>public static int AgeInYears(DateTime birthday, DateTime today)\n{\n   return ((today.Year - birthday.Year) * 372 + (today.Month - birthday.Month) * 31 + (today.Day - birthday.Day)) / 372;\n}\n</code></pre>\n This has been extensively unit-tested. It does look a bit "magic". The number 372 is the number of days there would be in a year if every month had 31 days. \n The explanation of why it works (<a href="https://social.msdn.microsoft.com/Forums/en-US/csharplanguage/thread/ba4a98af-aab3-4c59-bdee-611334e502f2"

rel="noreferrer">lifted from here) is: \n<blockquote>\n Let\'s set <code>Yn = DateTime.Now.Year, Yb = birthday.Year, Mn = DateTime.Now.Month, Mb = birthday.Month, Dn = DateTime.Now.Day, Db = birthday.Day</code> \n <code>age = Yn - Yb + (31*(Mn - Mb) + (Dn - Db)) / 372</code> \n We know that what we need is either <code>Yn-Yb</code> if the date has already been reached, <code>Yn-Yb-1</code> if it has not. \n a) If <code>Mn<Mb</code>, we have <code>-341 <= 31*(Mn-Mb) <= -31 and -30 <= Dn-Db <= 30</code> \n <code>-371 <= 31*(Mn - Mb) + (Dn - Db) <= -1</code> \n With integer division \n <code>(31*(Mn - Mb) + (Dn - Db)) / 372 = -1</code> \n b) If <code>Mn=Mb</code> and <code>Dn<Db</code>, we have <code>31*(Mn - Mb) = 0 and -30 <= Dn-Db <= -1</code> \n With integer division, again \n <code>(31*(Mn - Mb) + (Dn - Db)) / 372 = -1</code> \n c) If <code>Mn>Mb</code>, we have <code>31 <= 31*(Mn-Mb) <= 341 and -30 <= Dn-Db <= 30</code> \n <code>1 <= 31*(Mn - Mb) + (Dn - Db) <= 371</code> \n With integer division \n <code>(31*(Mn - Mb) + (Dn - Db)) / 372 = 0</code> \n d) If <code>Mn=Mb</code> and <code>Dn>Db</code>, we have <code>31*(Mn - Mb) = 0 and 1 <= Dn-Db <= 3</code>0 \n With integer division, again \n <code>(31*(Mn - Mb) + (Dn - Db)) / 372 = 0</code> \n e) If <code>Mn=Mb</code> and <code>Dn=Db</code>, we have <code>31*(Mn - Mb) + Dn-Db = 0</code> \n and therefore <code>(31*(Mn - Mb) + (Dn - Db)) / 372 = 0</code> \n</blockquote>\n The best way that I know of because of leap years and everything is: \n\n<pre class="lang-cs prettyprint-override"><code>DateTime birthDate = new DateTime(2000,3,1);\nint age = (int)Math.Floor((DateTime.Now - birthDate).TotalDays / 365.25D);\n</code></pre>\n Another function, not by me but found on the web and refined it a bit: \n\n<pre><code>public static int GetAge(DateTime birthDate)\n{\n   DateTime n = DateTime.Now; // To avoid a race condition around midnight\n   int age = n.Year - birthDate.Year;\n\n   if (n.Month < birthDate.Month || (n.Month == birthDate.Month && n.Day < birthDate.Day))\n   age--;\n\n   return age;\n}\n</code></pre>\n\n Just two things that come into my mind: What about people from countries that do not use the Gregorian calendar? DateTime.Now is in the server-specific culture I think. I have absolutely zero knowledge about actually working with Asian calendars and I do not know if there is an easy way to convert dates between calendars, but just in case you\'re wondering about those Chinese guys from the year 4660 :-) \n Simple Code \n\n<pre><code> var birthYear=1993;\n var age = DateTime.Now.AddYears(-birthYear).Year;\n</code></pre>\n === Common Saying (from months to years old) === \n\n If you just for common use, here is the code as your information: \n\n<pre><code>DateTime today = DateTime.Today;\nDateTime bday = DateTime.Parse("2016-2-14");\nint age = today.Year - bday.Year;\nvar unit = "";\n\nif (bday > today.AddYears(-age))\n{\n   age--;\n}\nif (age == 0)   // Under one year old\n{\n   age = today.Month - bday.Month;\n\n   age = age <= 0 ? (12 + age) : age; // The next year before birthday\n\n   age = today.Day - bday.Day >= 0 ? age : --age; // Before the birthday.day\n\n   unit = "month";\n}\nelse {\n   unit = "year";\n}\n\nif (age > 1)\n{\n

unit = unit + "s";\n}\n</code></pre>\n\n The test result as below: \n\n<pre><code>The birthday: 2016-2-14\n\n2016-2-15 => age=0, unit=month;\n2016-5-13 => age=2, unit=months;\n2016-5-14 => age=3, unit=months; \n2016-6-13 => age=3, unit=months; \n2016-6-15 => age=4, unit=months; \n2017-1-13 => age=10, unit=months; \n2017-1-14 => age=11, unit=months; \n2017-2-13 => age=11, unit=months; \n2017-2-14 => age=1, unit=year; \n2017-2-15 => age=1, unit=year; \n2017-3-13 => age=1, unit=year;\n2018-1-13 => age=1, unit=year; \n2018-1-14 => age=1, unit=year; \n2018-2-13 => age=1, unit=year; \n2018-2-14 => age=2, unit=years; \n</code></pre>\n<blockquote>\n How come the MSDN help did not tell you that? It looks so obvious: \n</blockquote>\n\n<pre class="lang-cs prettyprint-override"><code>System.DateTime birthTime = AskTheUser(myUser); // :-)\nSystem.DateTime now = System.DateTime.Now;\nSystem.TimeSpan age = now - birthTime; // As simple as that\ndouble ageInDays = age.TotalDays; // Will you convert to whatever you want yourself?\n</code></pre>\n I have used the following for this issue. I know it\'s not very elegant, but it\'s working. \n\n<pre class="lang-cs prettyprint-override"><code>DateTime zeroTime = new DateTime(1, 1, 1);\nvar date1 = new DateTime(1983, 03, 04);\nvar date2 = DateTime.Now;\nvar dif = date2 - date1;\nint years = (zeroTime + dif).Year - 1;\nLog.DebugFormat("Years -->{0}", years);\n</code></pre>\n<pre class="lang-cs prettyprint-override"><code>public string GetAge(this DateTime birthdate, string ageStrinFormat = null)\n{\n   var date = DateTime.Now.AddMonths(-birthdate.Month).AddDays(-birthdate.Day);\n   return string.Format(ageStrinFormat ?? "{0}/{1}/{2}",\n   (date.Year - birthdate.Year), date.Month, date.Day);\n}\n</code></pre>\n Keeping it simple (and possibly stupid:)). \n\n<pre class="lang-cs prettyprint-override"><code>DateTime birth = new DateTime(1975, 09, 27, 01, 00, 00, 00);\nTimeSpan ts = DateTime.Now - birth;\nConsole.WriteLine("You are approximately " + ts.TotalSeconds.ToString() + " seconds old.");\n</code></pre>\n To calculate the age with nearest age: \n\n<pre class="lang-cs prettyprint-override"><code>var ts = DateTime.Now - new DateTime(1988, 3, 19);\nvar age = Math.Round(ts.Days / 365.0);\n</code></pre>\n<pre><code>private int GetAge(int _year, int _month, int _day\n{\n   DateTime yourBirthDate= new DateTime(_year, _month, _day);\n\n   DateTime todaysDateTime = DateTime.Today;\n   int noOfYears = todaysDateTime.Year - yourBirthDate.Year;\n\n   if (DateTime.Now.Month < yourBirthDate.Month ||\n   (DateTime.Now.Month == yourBirthDate.Month && DateTime.Now.Day < yourBirthDate.Day))\n   {\n   noOfYears--;\n   }\n\n   return noOfYears;\n}\n</code></pre>\n This is the version we use here. It works, and it\'s fairly simple. It\'s the same idea as Jeff\'s but I think it\'s a little clearer because it separates out the logic for subtracting one, so it\'s a little easier to understand. \n\n<pre class="lang-cs prettyprint-override"><code>public static int GetAge(this DateTime dateOfBirth, DateTime dateAsAt)\n{\n   return dateAsAt.Year - dateOfBirth.Year - (dateOfBirth.DayOfYear < dateAsAt.DayOfYear ? 0 : 1);\n}\n</code></pre>\n\n You could expand the ternary operator to make it even clearer, if you think that sort of thing is unclear. \n\n Obviously

this is done as an extension method on <code>DateTime</code>, but clearly you can grab that one line of code that does the work and put it anywhere. Here we have another overload of the Extension method that passes in <code>DateTime.Now</code>, just for completeness. \n With fewer conversions and UtcNow, this code can take care of someone born on the Feb 29 in a leap year: \n\n<pre><code>public int GetAge(DateTime DateOfBirth)\n{\n   var Now = DateTime.UtcNow;\n   return Now.Year - DateOfBirth.Year -\n   (\n   (\n   Now.Month > DateOfBirth.Month ||\n   (Now.Month == DateOfBirth.Month && Now.Day >= DateOfBirth.Day)\n   ) ? 0 : 1\n   );\n}\n</code></pre>\n Here is the simplest way to calculate someone\'s age.
\nCalculating someone\'s age is pretty straightforward, and here\'s how! In order for the code to work, you need a DateTime object called BirthDate containing the birthday. \n\n<pre><code>C#\n   // get the difference in years\n   int years = DateTime.Now.Year - BirthDate.Year; \n   // subtract another year if we\'re before the\n   // birth day in the current year\n   if (DateTime.Now.Month < BirthDate.Month || \n   (DateTime.Now.Month == BirthDate.Month && \n   DateTime.Now.Day < BirthDate.Day))\n   years--;\n VB.NET\n   \' get the difference in years\n   Dim years As Integer = DateTime.Now.Year - BirthDate.Year\n   \' subtract another year if we\'re before the\n   \' birth day in the current year\n   If DateTime.Now.Month < BirthDate.Month Or (DateTime.Now.Month = BirthDate.Month And DateTime.Now.Day < BirthDate.Day) Then \n   years = years - 1\n   End If\n</code></pre>\n I would simply do this: \n\n<pre class="lang-cs prettyprint-override"><code>DateTime birthDay = new DateTime(1990, 05, 23);\nDateTime age = DateTime.Now - birthDay;\n</code></pre>\n\n This way you can calculate the exact age of a person, down to the millisecond if you want. \n<pre class="lang-cs prettyprint-override"><code>var birthDate = … // DOB\nvar resultDate = DateTime.Now - birthDate;\n</code></pre>\n\n Using <code>resultDate</code> you can apply <code>TimeSpan</code> properties whatever you want to display it. \n I think the TimeSpan has all that we need in it, without having to resort to 365.25 (or any other approximation). Expanding on Aug\'s example: \n\n<pre class="lang-cs prettyprint-override"><code>DateTime myBD = new DateTime(1980, 10, 10);\nTimeSpan difference = DateTime.Now.Subtract(myBD);\n\ntextBox1.Text = difference.Years + " years " + difference.Months + " Months " + difference.Days + " days";\n</code></pre>\n SQL version: \n\n<pre><code>declare @dd smalldatetime = \'1980-04-01\'\ndeclare @age int = YEAR(GETDATE())-YEAR(@dd)\nif (@dd> DATEADD(YYYY, -@age, GETDATE())) set @age = @age -1\n\nprint @age\n</code></pre>\n Here is a test snippet: \n\n<pre class="lang-cs prettyprint-override"><code>DateTime bDay = new DateTime(2000, 2, 29);\nDateTime now = new DateTime(2009, 2, 28);\nMessageBox.Show(string.Format("Test {0} {1} {2}",\n   CalculateAgeWrong1(bDay, now),   // outputs 9\n   CalculateAgeWrong2(bDay, now),   // outputs 9\n   CalculateAgeCorrect(bDay, now),   // outputs 8\n   CalculateAgeCorrect2(bDay, now))); // outputs 8\n</code></pre>\n\n Here you

have the methods: \n\n<pre class="lang-cs prettyprint-override"><code>public int CalculateAgeWrong1(DateTime birthDate, DateTime now)\n{\n   return new DateTime(now.Subtract(birthDate).Ticks).Year - 1;\n}\n\npublic int CalculateAgeWrong2(DateTime birthDate, DateTime now)\n{\n   int age = now.Year - birthDate.Year;\n\n   if (now < birthDate.AddYears(age))\n age--;\n\n   return age;\n}\n\npublic int CalculateAgeCorrect(DateTime birthDate, DateTime now)\n{\n   int age = now.Year - birthDate.Year;\n\n   if (now.Month < birthDate.Month || (now.Month == birthDate.Month && now.Day < birthDate.Day))\n   age--;\n\n   return age;\n}\n\npublic int CalculateAgeCorrect2(DateTime birthDate, DateTime now)\n{\n   int age = now.Year - birthDate.Year;\n\n   // For leap years we need this\n   if (birthDate > now.AddYears(-age)) \n   age--;\n   // Don\'t use:\n   // if (birthDate.AddYears(age) > now) \n   //   age--;\n\n   return age;\n}\n</code></pre>\n I have created a SQL Server User Defined Function to calculate someone\'s age, given their birthdate. This is useful when you need it as part of a query: \n\n<pre><code>using System;\nusing System.Data;\nusing System.Data.Sql;\nusing System.Data.SqlClient;\nusing System.Data.SqlTypes;\nusing Microsoft.SqlServer.Server;\n\npublic partial class UserDefinedFunctions\n{\n   [SqlFunction(DataAccess = DataAccessKind.Read)]\n public static SqlInt32 CalculateAge(string strBirthDate)\n   {\n DateTime dtBirthDate = new DateTime();\n   dtBirthDate = Convert.ToDateTime(strBirthDate);\n   DateTime dtToday = DateTime.Now;\n\n // get the difference in years\n   int years = dtToday.Year - dtBirthDate.Year;\n\n   // subtract another year if we\'re before the\n // birth day in the current year\n   if (dtToday.Month < dtBirthDate.Month || (dtToday.Month == dtBirthDate.Month && dtToday.Day < dtBirthDate.Day))\n   years=years-1;\n\n   int intCustomerAge = years;\n   return intCustomerAge;\n   }\n};\n</code></pre>\n<pre class="lang-cs prettyprint-override"><code>TimeSpan diff = DateTime.Now - birthdayDateTime;\nstring age = String.Format("{0:%y} years, {0:%M} months, {0:%d}, days old", diff);\n</code></pre>\n\n I\'m not sure how exactly you\'d like it returned to you, so I just made a readable string. \n The simplest way I\'ve ever found is this. It works correctly for the US and western europe locales. Can\'t speak to other locales, especially places like China. 4 extra compares, at most, following the initial computation of age. \n\n<pre><code>public int AgeInYears(DateTime birthDate, DateTime referenceDate)\n{\n Debug.Assert(referenceDate >= birthDate, \n "birth date must be on or prior to the reference date");\n\n DateTime birth = birthDate.Date;\n DateTime reference = referenceDate.Date;\n int years = (reference.Year - birth.Year);\n\n //\n // an offset of -1 is applied if the birth date has \n // not yet occurred in the current year.\n //\n if (reference.Month > birth.Month);\n else if (reference.Month < birth.Month) \n   --years;\n else // in birth month\n {\n   if (reference.Day < birth.Day)\n   --years;\n }\n\n return years;\n}\n</code></pre>\n\n I was looking over the answers to this and noticed that nobody has made reference to regulatory/legal implications of leap day births. For instance, <a href="https://en.wikipedia.org/wiki/February_29#Births"

```
rel="nofollow noreferrer">per Wikipedia</a>, if you\'re born on February 29th in
various jurisdictions, you\'re non-leap year birthday
varies:</p>\n\n<ul>\n<li>In the United Kingdom and Hong Kong: it\'s the ordinal
day of the year, so the next day, March 1st is your birthday.</li>\n<li>In New
Zealand: it\'s the previous day, February 28th for the purposes of driver
licencing, and March 1st for other purposes.</li>\n<li>Taiwan: it\'s February
28th.</li>\n</ul>\n\n<p>And as near as I can tell, in the US, the statutes are
silent on the matter, leaving it up to the common law and to how various
regulatory bodies define things in their regulations.</p>\n\n<p>To that end, an
improvement:</p>\n\n<pre><code>public enum LeapDayRule\n{\n  OrdinalDay     = 1
,\n  LastDayOfMonth = 2 ,\n}\n\nstatic int ComputeAgeInYears(DateTime birth,
DateTime reference, LeapYearBirthdayRule ruleInEffect)\n{\n  bool
isLeapYearBirthday = CultureInfo.CurrentCulture.Calendar.IsLeapDay(birth.Year,
birth.Month, birth.Day);\n  DateTime cutoff;\n\n  if (isLeapYearBirthday
&amp;amp;&amp;amp; !DateTime.IsLeapYear(reference.Year))\n  {\n    switch
(ruleInEffect)\n    {\n      case LeapDayRule.OrdinalDay:\n        cutoff = new
DateTime(reference.Year, 1, 1)\n
.AddDays(birth.DayOfYear - 1);\n        break;\n\n      case
LeapDayRule.LastDayOfMonth:\n        cutoff = new DateTime(reference.Year,
birth.Month, 1)\n                                .AddMonths(1)\n
.AddDays(-1);\n        break;\n\n      default:\n        throw new
InvalidOperationException();\n    }\n  }\n  else\n  {\n    cutoff = new
DateTime(reference.Year, birth.Month, birth.Day);\n  }\n\n  int age =
(reference.Year - birth.Year) + (reference &gt;= cutoff ? 0 : -1);\n  return age
&lt; 0 ? 0 : age;\n}\n</code></pre>\n\n<p>It should be noted that this code
assumes:</p>\n\n<ul>\n<li>A western (European) reckoning of age, and</li>\n<li>A
calendar, like the Gregorian calendar that inserts a single leap day at the end
of a month.</li>\n</ul>\n<p>I use this:</p>\n\n<pre><code>public static class
DateTimeExtensions\n{\n    public static int Age(this DateTime birthDate)\n
{\n        return Age(birthDate, DateTime.Now);\n    }\n\n    public static int
Age(this DateTime birthDate, DateTime offsetDate)\n    {\n        int
result=0;\n        result = offsetDate.Year - birthDate.Year;\n\n        if
(offsetDate.DayOfYear &lt; birthDate.DayOfYear)\n        {\n
result--;\n        }\n\n        return result;\n    }\n}\n</code></pre>\n<p>I
want to add Hebrew calendar calculations (or other System.Globalization calendar
can be used in the same way),  using rewrited functions from this
thread:</p>\n\n<pre class="lang-cs prettyprint-override"><code>Public Shared
Function CalculateAge(BirthDate As DateTime) As Integer\n    Dim HebCal As New
System.Globalization.HebrewCalendar ()\n    Dim now = DateTime.Now()\n    Dim
iAge = HebCal.GetYear(now) - HebCal.GetYear(BirthDate)\n    Dim iNowMonth =
HebCal.GetMonth(now), iBirthMonth = HebCal.GetMonth(BirthDate)\n    If iNowMonth
&lt; iBirthMonth Or (iNowMonth = iBirthMonth AndAlso HebCal.GetDayOfMonth(now)
&lt; HebCal.GetDayOfMonth(BirthDate)) Then iAge -= 1\n    Return iAge\nEnd
Function\n</code></pre>'
```

[ ]:
```
preprocessed_answers[0]
```

[ ]: 'following approach extract time period library net class datediff considers calendar culture info usage solution suggestion seems year changing right date spot tested age simple answer simple easy follow example use current date get total hours divide total hours per year get exactly age months days may work datetime extender adds age calculation datetime object one accurate answers able resolve birthday th feb compared year th feb need consider people smaller year chinese culture describe small babies age months weeks implementation simple imagined especially deal date like implementation passed test cases hope helpful think top answer clear often count fingers need look calendar work things change would code running linqpad gives code linqpad easiest way answer single line also works leap years wow give answer many answers simple question made one small change mark soen answer rewriten third line expression parsed bit easily also made function sake clarity spent time working came calculate someone age years months days tested feb th problem leap years seems work would appreciate feedback classic question deserving noda time solution usage might also interested following improvements passing clock instead using would improve testability target time zone likely change would want parameter well see also blog post subject handling birthdays anniversaries easy understand simple solution however assumes looking western idea age using east asian reckoning one liner answer calculate many years old person little code sample c# knocked careful around edge cases specifically leap years solutions take account pushing answer datetime cause problems could end trying put many days specific month e g days feb simple appears accurate needs making assumption purpose leap years regardless person chooses celebrate birthday technically year older days passed since last birthday e th february make year older let us know spot problems check simple answer apply shown native method add years th feb leap years obtain correct result th feb common years feel th mar birthday leaplings neither net official rule supports common logic explain born february birthdays another month age method lends added extension obtain age simplest possible way list item int age birthdate age run test critical date example birth date later date age output later date main problems solve calculate exact age years months days etc calculate generally perceived age people usually care old exactly care birthday current year solution obvious solution one precise determing total age perceived precise people people also usually use calculate age manually notes preferred solution cannot use datetime dayofyear timespans shift number days leap years put little lines readability one note would create static overloaded methods one universal usage second usage friendliness one liner simple strange way format date subtract date birth current date drop last digits got age know c# believe work language drop last digits c# code alternatively without type conversion form extension method error checking omitted gives detail question maybe looking customized method calculate age plus bonus validation message case helps method call pass datetime value mm dd yyyy server set usa locale replace anything messagebox container display remember format message way like try solution working created age struct looks like function serving well calculations simple unit test used scarcher solution accurate year calculation persons age needed take calculate months days along years direct answer philosophical reasoning problem hand quasi scientific point view would argue

```
question specify unit culture measure age answers seem assume integer annual
representation si unit time ergo correct generic answer course assuming
normalized taking regard whatsoever relativistic effects christian way
calculating age years finance similar problem calculating something often
referred day count fraction roughly number years given period age issue really
time measuring issue example actual actual counting days correctly convention
another quite common way measure time generally serializing dude named date
convention must seriously trippin wonder long go relativistic age seconds
becomes useful rough approximation earth around sun cycles one lifetime far
words period must given location function representing motion valid solution yet
another answer extensively unit tested look bit quot magic quot number number
days would year every month days explanation works lifted let set know need
either date already reached integer division b integer division c integer
division integer division e therefore best way know leap years everything
another function found web refined bit two things come mind people countries use
gregorian calendar datetime server specific culture think absolutely zero
knowledge actually working asian calendars know easy way convert dates calendars
case wondering chinese guys year simple code common saying months years old
common use code information test result come msdn help tell looks obvious used
following issue know elegant working keeping simple possibly stupid calculate
age nearest age version use works fairly simple idea jeff think little clearer
separates logic subtracting one little easier understand could expand ternary
operator make even clearer think sort thing unclear obviously done extension
method clearly grab one line code work put anywhere another overload extension
method passes completeness fewer conversions utcnow code take care someone born
feb leap year simplest way calculate someone age calculating someone age pretty
straightforward order code work need datetime object called birthdate containing
birthday would simply way calculate exact age person millisecond want using
apply properties whatever want display think timespan need without resort
approximation expanding aug example sql version test snippet methods created sql
server user defined function calculate someone age given birthdate useful need
part query sure exactly would like returned made readable string simplest way
ever found works correctly us western europe locales speak locales especially
places like china extra compares following initial computation age looking
answers noticed nobody made reference regulatory legal implications leap day
births instance per wikipedia born february th various jurisdictions non leap
year birthday varies united kingdom hong kong ordinal day year next day march st
birthday new zealand previous day february th purposes driver licencing march st
purposes taiwan february th near tell us statutes silent matter leaving common
law various regulatory bodies define things regulations end improvement noted
code assumes western european reckoning age calendar like gregorian calendar
inserts single leap day end month use want add hebrew calendar calculations
system globalization calendar used way using rewrited functions thread '
```

[ ]:
```
# dump preprocessed question drive
joblib.dump(preprocessed_questions,'/content/drive/My Drive/self_case_study1/
 ↪data/preprocessed_questions')
```

[ ]: ['/content/drive/My Drive/self_case_study1/data/preprocessed_questions']

[ ]:
```
# dump preprocessed title in drive
joblib.dump(preprocessed_title,'/content/drive/My Drive/self_case_study1/data/
 ↪preprocessed_title')
```

[ ]: ['/content/drive/My Drive/self_case_study1/data/preprocessed_title']

[ ]:
```
# dump preprocessed answers in drive
joblib.dump(preprocessed_answers,'/content/drive/My Drive/self_case_study1/data/
 ↪preprocessed_answers')
```

[ ]: ['/content/drive/My Drive/self_case_study1/data/preprocessed_answers']

[ ]:
```
# load preprocessed questions
preprocessed_question = joblib.load('/content/drive/My Drive/self_case_study1/
 ↪data/preprocessed_questions')
```

[ ]:
```
print('Length of preprocessed questions = ',len(preprocessed_question))
```

```
Length of preprocessed questions =  572554
```

[ ]:
```
# load preprocessed titles
preprocessed_title = joblib.load('/content/drive/My Drive/self_case_study1/data/
 ↪preprocessed_title')
```

[ ]:
```
print('Length of preprocessed titles= ',len(preprocessed_title))
```

```
Length of preprocessed titles=  572554
```

[ ]:
```
# load preprocessed answers
preprocessed_answer = joblib.load('/content/drive/My Drive/self_case_study1/
 ↪data/preprocessed_answers')
```

[ ]:
```
print('Length of preprocessed answers= ',len(preprocessed_answer))
```

```
Length of preprocessed answers=  572554
```

[ ]:
```
tags = df['tags']
id = df['id']
score = df['score']
```

[ ]:
```
Dict = { 'id' : id,
         'preprocessed_title' : preprocessed_title,
         'preprocessed_question': preprocessed_question,
         'preprocessed_answer' : preprocessed_answer,
         'score' : score,
         'tags': tags}
```

```
preprocessed_df = pd.DataFrame(Dict)
```

[ ]:
```
# save as csv
# preprocessed_df.to_csv('/content/drive/My Drive/self_case_study1/data/
 ↪preprocessed_df.csv',index=False)
```

0.1 Load preprocesed data

[ ]:
```
preprocessed_df = pd.read_csv('/content/drive/My Drive/self_case_study1/data/
 ↪preprocessed_df.csv')
preprocessed_df.head()
```

[ ]:
```
    id  …                                          tags
0    9  …                                c#|.net|datetime
1   11  …  c#|datetime|time|datediff|relative-time-span
2  126  …                           java|php|oop|theory
3  330  …                  c++|oop|class|nested-class
4  742  …                     python|django|views|oop

[5 rows x 6 columns]
```

[ ]:
```
# cheking for null values
print('Total null values in preprocessed titles␣
 ↪=',preprocessed_df['preprocessed_title'].isna().sum())
print('Total null values in preprocessed questions␣
 ↪=',preprocessed_df['preprocessed_question'].isna().sum())
print('Total null values in preprocessed answers␣
 ↪=',preprocessed_df['preprocessed_answer'].isna().sum())
print('Total null values in score =',preprocessed_df['score'].isna().sum())
print('Total null values in tags =',preprocessed_df['tags'].isna().sum())
```

```
Total null values in preprocessed titles = 5
Total null values in preprocessed questions = 390
Total null values in preprocessed answers = 3436
Total null values in score = 0
Total null values in tags = 1
```

[ ]:
```
# concatenate title and question body
preprocessed_df['question_content'] = preprocessed_df['preprocessed_title'].
 ↪fillna('') + ' ' + preprocessed_df['preprocessed_question'].fillna('')
```

[ ]:
```
# remove all character  which has length 1
def remove_single_char(text):
    l = [word for word in str(text).split() if len(word) > 1]
    return ' '.join(l)
```

```
[ ]: # remove words with length 1 from question content and answer
preprocessed_df['question_content'] = preprocessed_df['question_content'].
 ↪apply(lambda x: remove_single_char(x))
preprocessed_df['preprocessed_answer'] = preprocessed_df['preprocessed_answer'].
 ↪apply(lambda x: remove_single_char(x))
```

```
[ ]: # concatenate question content and preprocessed answer
preprocessed_df['all_text'] = preprocessed_df['question_content'].fillna('')+'␣
 ↪'+preprocessed_df['preprocessed_answer'].fillna('')
```

```
[ ]: preprocessed_df.head()
```

```
[ ]:     id  …                                              all_text
     0    9  …  c# calculate someone age based datetime type b…
     1   11  …  calculate relative time c# given specific valu…
     2  126  …  would access object properties within object m…
     3  330  …  use nested classes case working collection cla…
     4  742  …  class views django django view points function…

     [5 rows x 8 columns]
```

Checking for null values in question content

```
[ ]: # checking in preprocessed_df for null titles
preprocessed_df['question_content'].isna().sum()
```

```
[ ]: 0
```

Checking for null values in all text

```
[ ]: # checking in preprocessed_df for null answers
preprocessed_df['all_text'].isna().sum()
```

```
[ ]: 0
```

```
[ ]: preprocessed_df['all_text'][0]
```

```
[ ]: 'c# calculate someone age based datetime type birthday given type representing
     person birthday calculate age years following approach extract time period
     library net class datediff considers calendar culture info usage solution
     suggestion seems year changing right date spot tested age simple answer simple
     easy follow example use current date get total hours divide total hours per year
     get exactly age months days may work datetime extender adds age calculation
     datetime object one accurate answers able resolve birthday th feb compared year
     th feb need consider people smaller year chinese culture describe small babies
     age months weeks implementation simple imagined especially deal date like
     implementation passed test cases hope helpful think top answer clear often count
```

fingers need look calendar work things change would code running linqpad gives code linqpad easiest way answer single line also works leap years wow give answer many answers simple question made one small change mark soen answer rewriten third line expression parsed bit easily also made function sake clarity spent time working came calculate someone age years months days tested feb th problem leap years seems work would appreciate feedback classic question deserving noda time solution usage might also interested following improvements passing clock instead using would improve testability target time zone likely change would want parameter well see also blog post subject handling birthdays anniversaries easy understand simple solution however assumes looking western idea age using east asian reckoning one liner answer calculate many years old person little code sample c# knocked careful around edge cases specifically leap years solutions take account pushing answer datetime cause problems could end trying put many days specific month days feb simple appears accurate needs making assumption purpose leap years regardless person chooses celebrate birthday technically year older days passed since last birthday th february make year older let us know spot problems check simple answer apply shown native method add years th feb leap years obtain correct result th feb common years feel th mar birthday leaplings neither net official rule supports common logic explain born february birthdays another month age method lends added extension obtain age simplest possible way list item int age birthdate age run test critical date example birth date later date age output later date main problems solve calculate exact age years months days etc calculate generally perceived age people usually care old exactly care birthday current year solution obvious solution one precise determing total age perceived precise people people also usually use calculate age manually notes preferred solution cannot use datetime dayofyear timespans shift number days leap years put little lines readability one note would create static overloaded methods one universal usage second usage friendliness one liner simple strange way format date subtract date birth current date drop last digits got age know c# believe work language drop last digits c# code alternatively without type conversion form extension method error checking omitted gives detail question maybe looking customized method calculate age plus bonus validation message case helps method call pass datetime value mm dd yyyy server set usa locale replace anything messagebox container display remember format message way like try solution working created age struct looks like function serving well calculations simple unit test used scarcher solution accurate year calculation persons age needed take calculate months days along years direct answer philosophical reasoning problem hand quasi scientific point view would argue question specify unit culture measure age answers seem assume integer annual representation si unit time ergo correct generic answer course assuming normalized taking regard whatsoever relativistic effects christian way calculating age years finance similar problem calculating something often referred day count fraction roughly number years given period age issue really time measuring issue example actual actual counting days correctly convention another quite common way measure time generally serializing dude named date convention must seriously trippin wonder long go relativistic age seconds becomes useful rough approximation earth around sun cycles one lifetime far

words period must given location function representing motion valid solution yet another answer extensively unit tested look bit quot magic quot number number days would year every month days explanation works lifted let set know need either date already reached integer division integer division integer division integer division therefore best way know leap years everything another function found web refined bit two things come mind people countries use gregorian calendar datetime server specific culture think absolutely zero knowledge actually working asian calendars know easy way convert dates calendars case wondering chinese guys year simple code common saying months years old common use code information test result come msdn help tell looks obvious used following issue know elegant working keeping simple possibly stupid calculate age nearest age version use works fairly simple idea jeff think little clearer separates logic subtracting one little easier understand could expand ternary operator make even clearer think sort thing unclear obviously done extension method clearly grab one line code work put anywhere another overload extension method passes completeness fewer conversions utcnow code take care someone born feb leap year simplest way calculate someone age calculating someone age pretty straightforward order code work need datetime object called birthdate containing birthday would simply way calculate exact age person millisecond want using apply properties whatever want display think timespan need without resort approximation expanding aug example sql version test snippet methods created sql server user defined function calculate someone age given birthdate useful need part query sure exactly would like returned made readable string simplest way ever found works correctly us western europe locales speak locales especially places like china extra compares following initial computation age looking answers noticed nobody made reference regulatory legal implications leap day births instance per wikipedia born february th various jurisdictions non leap year birthday varies united kingdom hong kong ordinal day year next day march st birthday new zealand previous day february th purposes driver licencing march st purposes taiwan february th near tell us statutes silent matter leaving common law various regulatory bodies define things regulations end improvement noted code assumes western european reckoning age calendar like gregorian calendar inserts single leap day end month use want add hebrew calendar calculations system globalization calendar used way using rewrited functions thread'

1 Analysis of question content

[ ]:
```
# import FreqDist from nltk library to get word frequency from given text
from nltk import FreqDist
```

[ ]:
```
# get one single list of whole corpus words
all_words = []
for line in tqdm(preprocessed_df['question_content']):
    all_words.extend(line.split())
```

```
HBox(children=(FloatProgress(value=0.0, max=572554.0), HTML(value='')))
```

[ ]:
```
# get frequency distribution of all words
word_freq = FreqDist(all_words)
print('Number of unique words = ',len(word_freq))
```

```
Number of unique words =  356348
```

[ ]:
```
# cheecking last 5 words from most common words
word_freq.most_common(20000)[-5:]
```

[ ]:
```
[('collectstatic', 30),
 ('cloudsearch', 30),
 ('pingfederate', 30),
 ('isnumber', 30),
 ('crl', 30)]
```

here we are taking top 20000 words because last 5 words of top 20000 words are occuring atleasr 30 times

[ ]:
```
# top 20000 words
top_k_words = dict(word_freq.most_common(20000)).keys()
```

[ ]:
```
# set of top k words
top_k_words = set(top_k_words)
print('Length of top k words is = ', len(top_k_words))
```

```
Length of top k words is =  20000
```

[ ]:
```
def keep_top_k_words(text):
    ''' This function will keep only top k words'''
    top_k =  [word for word in text.split() if word in top_k_words]
    return ' '.join(top_k)
```

[ ]:
```
# apply keep top words function on question content
preprocessed_df['question_content'] = preprocessed_df['question_content'].
 ↪apply(lambda x : keep_top_k_words(x))
```

1.0.1 Analysis on question content length

[ ]:
```
# make column of question content length
preprocessed_df['question_length'] = preprocessed_df['question_content'].
 ↪apply(lambda x: len(x.split()))
```

[ ]:
```
preprocessed_df.head()
```

[ ]:
```
     id  … question_length
0     9  …               16
1    11  …               17
2   126  …               96
3   330  …               82
4   742  …               56

[5 rows x 9 columns]
```

[ ]:
```
preprocessed_df.shape
```

[ ]:
```
(572554, 9)
```

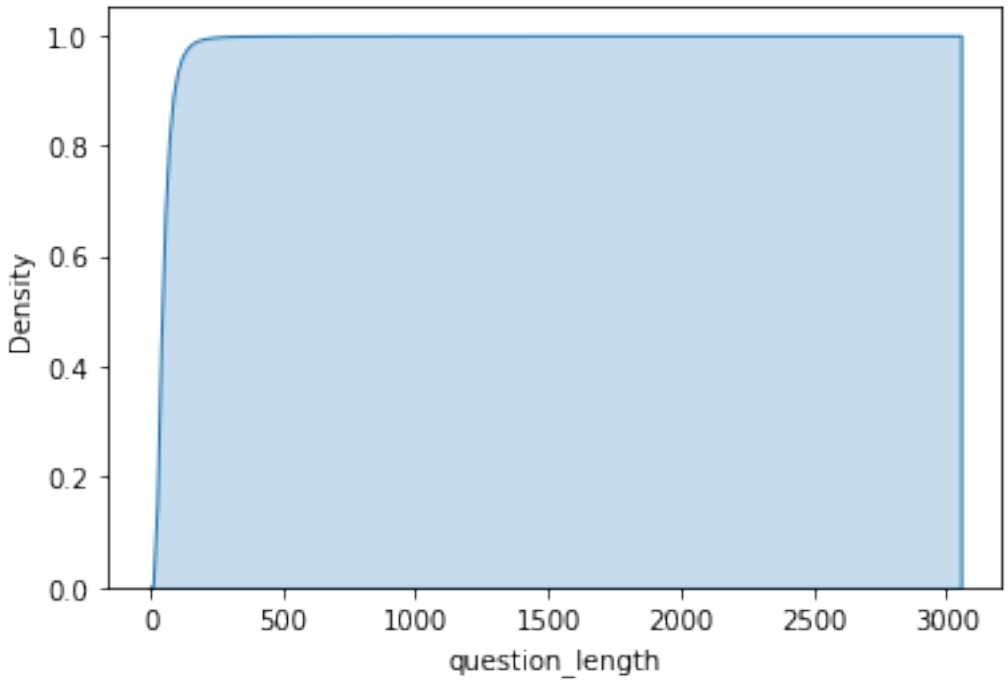
[ ]:
```
print('50th percentile of question lenght is ==>',np.
 ↪percentile(preprocessed_df['question_length'],50))
print('75th percentile of question lenght is ==>',np.
 ↪percentile(preprocessed_df['question_length'],75))
print('90th percentile of question lenght is ==>',np.
 ↪percentile(preprocessed_df['question_length'],90))
print('95th percentile of question lenght is ==>',np.
 ↪percentile(preprocessed_df['question_length'],95))
print('98th percentile of question lenght is ==>',np.
 ↪percentile(preprocessed_df['question_length'],98))
```

```
50th percentile of question lenght is ==> 41.0
75th percentile of question lenght is ==> 60.0
90th percentile of question lenght is ==> 87.0
95th percentile of question lenght is ==> 109.0
98th percentile of question lenght is ==> 143.0
```

1.1 plot CDF of question length

[ ]:
```
import seaborn as sns
sns.kdeplot(preprocessed_df['question_length'],cumulative=True,shade=True)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcfb769c898>
```



1.1.1 as we can see 75th percent of question length are 75 or less then 75 so we will consider all questions which have length less then or equal to 75.

[ ]:
```
print('Total number of rows which have question length greater then 60 is ␣
↪=======>')
print(preprocessed_df[preprocessed_df['question_length']>60].shape[0])
```

```
Total number of rows which have question length greater then 60 is  =======>
84586
```

[ ]:
```
preprocessed_df = preprocessed_df[preprocessed_df['question_length']<60]
print('Total number of rows in preprocessed_df after dropping large length␣
↪question = ', preprocessed_df.shape[0])
```

```
Total number of rows in preprocessed_df after dropping large length question =
425167
```

[ ]:
```
# reset index
preprocessed_df.reset_index(drop=True,inplace=True)
```

[ ]:
```
preprocessed_df.head()
```

[ ]:
```
      id  … question_length
0      9  …               16
1     11  …               17
2    742  …               56
3   2214  …               23
4   2509  …               20

[5 rows x 9 columns]
```

Save featurized data as csv

[ ]:
```
# save final featurized dataframe
preprocessed_df.to_csv('/content/drive/My Drive/self_case_study1/data/
 ↪featurized_df.csv',index=False)
```

Load featurized dataframe

[ ]:
```
featurized_df = pd.read_csv('/content/drive/My Drive/self_case_study1/data/
 ↪featurized_df.csv')
```

[ ]:
```
print('Shape of featurized_df is ===> ',featurized_df.shape)
```

```
Shape of featurized_df is ===>  (425167, 9)
```

[ ]:
```
featurized_df.head(3)
```

[ ]:
```
     id  … question_length
0     9  …               16
1    11  …               17
2   742  …               56

[3 rows x 9 columns]
```

[ ]:
```
from sklearn.model_selection import train_test_split
X_train,X_test = train_test_split(featurized_df,test_size=0.5,random_state=42)

# reset index of train and test
X_train.reset_index(drop=True,inplace=True)
X_test.reset_index(drop=True,inplace=True)

print('Shape of X_train',X_train.shape)
print('Shape of X_test',X_test.shape)
```

```
Shape of X_train (212583, 9)
Shape of X_test (212584, 9)
```

[ ]:
```
X_train.head()
```

[ ]:
```
          id  … question_length
0   63161427  …               47
1   61920795  …               28
2   61402494  …               21
3   28564564  …               39
4    3785444  …               13

[5 rows x 9 columns]
```

1.1.2 TFIDF word2vec using pretrained glove model

[ ]:
```
# function to load glove vectors
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('/content/drive/My Drive/Glove_42B/glove.42B.300d.txt')
```

```
Loading Glove Model
```

```
HBox(children=(FloatProgress(value=1.0, bar_style='info', max=1.0), HTML(value='')))
```

```
Done. 1917494  words loaded!
```

[ ]:
```
# get list of lists for every sentence words
words = []
for line in tqdm(X_train['question_content']):
    words.append(line.split())
```

```
HBox(children=(FloatProgress(value=0.0, max=212583.0), HTML(value='')))
```

[ ]:
```
len(words)
```

[ ]:
```
212583
```

[ ]:
```
# get one single list of whole corpus words
Words = []
for line in tqdm(X_train['question_content']):
    Words.extend(line.split())
```

```
HBox(children=(FloatProgress(value=0.0, max=212583.0), HTML(value='')))
```

[ ]:
```
print("All the words in the courpus", len(Words))
Words = set(Words)
print("The unique words in the courpus", len(Words))
inter_words = set(model.keys()).intersection(Words)
print("The number of words that are present in both glove vectors and our␣
 ↪courpus", \
len(inter_words),"(",np.round(len(inter_words)/len(Words)*100,3),"%)")
```

```
All the words in the courpus 7340998
The unique words in the courpus 19725
The number of words that are present in both glove vectors and our courpus 18546
( 94.023 %)
```

[ ]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(min_df=10)
# fit and transform on question content
tfidf.fit_transform(X_train["question_content"])
```

[ ]:
```
<212583x14175 sparse matrix of type '<class 'numpy.float64'>'
        with 5440200 stored elements in Compressed Sparse Row format>
```

[ ]:
```
# save tfidf
joblib.dump(tfidf,'/content/drive/My Drive/self_case_study1/data/
 ↪joblib_dumps_glove_model/tfidf1')
```

[ ]:
```
['/content/drive/My
Drive/self_case_study1/data/joblib_dumps_glove_model/tfidf1']
```

[ ]:
```
len(tfidf_words)
```

[ ]:
```
14175
```

[ ]:
```
words_courpus = {}
words_glove = set(model.keys())
for i in Words:
    if i in words_glove:
        words_courpus[i] = model[i]
```

```
print("word 2 vec length", len(words_courpus))
# stronging variables into pickle files python: http://www.jessicayung.com/
 ↪how-to-use-pickle-to-save-and-load-variables-in-python/

import pickle
with open('/content/drive/My Drive/self_case_study1/data/
 ↪joblib_dumps_glove_model/glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)
```

```
word 2 vec length 18546
```

1.2 load glove vectors

[129]:
```
with open('/content/drive/My Drive/self_case_study1/data/
 ↪joblib_dumps_glove_model/glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words = set(model.keys())
```

[130]:
```
# we are converting a dictionary with word as a key, and the idf as a value
tfidf = joblib.load('/content/drive/My Drive/self_case_study1/data/
 ↪joblib_dumps_glove_model/tfidf1')
dictionary = dict(zip(tfidf.get_feature_names(), list(tfidf.idf_)))
tfidf_words = set(tfidf.get_feature_names())
```

[ ]:
```
tfidf_w2v_vector = [] # the weighted w2v for each sentence is stored in this list
for sentence in tqdm(words): # for each sentence words list
    vector = np.zeros(300) # as word vectors length is 300
    tf_idf_weight =0; # num of words with a valid vector in the sentence
    for word in sentence : # for each word in a review/sentence
        if (word in glove_words ) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # multiply idf value of word with tf of word
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence))
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vector.append(vector)
```

```
HBox(children=(FloatProgress(value=0.0, max=212583.0), HTML(value='')))
```

[ ]:
```
# save tfidf w2v vectors
```

```
joblib.dump(tfidf_w2v_vector,'/content/drive/My Drive/self_case_study1/data/
↪joblib_dumps_glove_model/tfidf_w2v_vector_glove')
```

[ ]:
```
['/content/drive/My
Drive/self_case_study1/data/joblib_dumps_glove_model/tfidf_w2v_vector_glove']
```

[125]:
```
# load tfidf w2v vectors
tfidf_w2v_vector = joblib.load('/content/drive/My Drive/self_case_study1/data/
↪joblib_dumps_glove_model/tfidf_w2v_vector_glove')
```

[121]:
```
# query question
text = 'how to make dictionary in python language'
text
```

[121]:
```
'how to make dictionary in python language'
```

[131]:
```
# get glove words
#glove_words = set(model.keys())
# get tfidf words
#tfidf_words = set(tfidf.get_feature_names())
# dictionary of tfidf
#dictionary = dict(zip(tfidf.get_feature_names(), list(tfidf.idf_)))


def find_similarity(question,top_n):
    ''' This function will find top similar result for given query'''
    start = time.time()
    # initialize  vector for user query
    main_vec = np.zeros(300)
    # initialize tfidf weight
    weight_sum = 0
    # preprocess question
    text = preprocess_title(question)
    #splitting the sentence
    text_list = list(text.split())
    for word in text_list:
        #finding if word is present in tfidf and in w2v words
        if word in tfidf_words and word in glove_words :
            #finding vector of word from glove model
            vect = model[word]
            #compute tfidf
            tf_idf = dictionary[word]*(text_list.count(word)/len(text_list))
            # adding vector * tfidf to main_vec
            main_vec+= (vect*tf_idf)
            # summing tfidf values
            weight_sum += tf_idf
    if weight_sum !=0:
```

```
        # devide by weight_sum
        main_vec /= weight_sum
    # find cosine similarity
    similarities =  cosine_similarity((main_vec).reshape(1, -1),␣
↪Y=tfidf_w2v_vector, dense_output=True)
    # sort similarities
    #print(similarities[0])

    sort = np.argsort(similarities[0])
    # get top similarity indices  in descending order
    similarity_index = np.array(list(reversed(sort)))
    # finad top n similarities
    top_similarity_index = similarity_index[:top_n]
    # print top similarity values
    print('Top cosine similarities are␣
↪======>',similarities[0][top_similarity_index])
    similar_questions = X_train['preprocessed_title'][top_similarity_index]
    #print(similar_questions)

    similar_question_list = []
    for q in similar_questions:
        similar_question_list.append(q)
        #print(q)
    total_time = (time.time() - start)
    print('Total time ===========> ',total_time)
    return similar_question_list
```

[132]:
```
top_10_questions = find_similarity(text,10)
```

```
Top cosine similarities are ======> [0.91009833 0.90754653 0.90387262 0.8991957
0.8973932  0.89601402
 0.89469953 0.89280858 0.8910764  0.88964025]
Total time ===========>  1.557215690612793
```

[133]:
```
print('Top 10 similar questions Using weighted TFIDF and Glove vectors')
print('='*100)
print('\t')
for i,line in enumerate(top_10_questions):
    print('Question {} ===> {}'.format(i+1,line))
    print('='*100)
```

```
Top 10 similar questions Using weighted TFIDF and Glove vectors
====================================================================================
====================

Question 1 ===> handling dictionary object python
====================================================================================
```

```
====================
Question 2 ===> need help translating language function wordpress
================================================================================
====================
Question 3 ===> creating words game java issues dictionary file organisation
================================================================================
====================
Question 4 ===> iteration nested dictionary python
================================================================================
====================
Question 5 ===> expressjs language library
================================================================================
====================
Question 6 ===> create dictionary within another dictionary count capability
================================================================================
====================
Question 7 ===> possible reverse dictionary python using dictionary
comprehension
================================================================================
====================
Question 8 ===> wpf spell check specific language
================================================================================
====================
Question 9 ===> change android app language without changing phone language
================================================================================
====================
Question 10 ===> import nested dictionary pandas yaml
================================================================================
====================
```

2 train word2vec using gensim

[ ]:
```
from gensim.models import Word2Vec
```

[ ]:
```
# train Word2vec model with vector size 60
w2v_model = Word2Vec(words,size=60,min_count=10, workers=-1,iter=50)
```

[ ]:
```
word_vectors = w2v_model.wv
```

[ ]:
```
# shape of w2v vectors
word_vectors.vectors.shape
```

[ ]:
```
(16226, 60)
```

[ ]:
```
# save word2vec model
joblib.dump(w2v_model,'/content/drive/My Drive/self_case_study1/data/
↪joblib_dumps/gensim_w2v_model')
```

[ ]:
```
['/content/drive/My Drive/self_case_study1/data/joblib_dumps/gensim_w2v_model']
```

[ ]:
```
# get word2vec vocab
w2v_words = list(w2v_model.wv.vocab)
print("sample words =====>", w2v_words[0:10])
```

```
sample words =====> ['react', 'navigation', 'stack', 'navigator', 'using',
'app', 'tab', 'contains', 'want', 'use']
```

[ ]:
```
# number of unique word in word2vec vocab
print('Number of unique words in Word2vec ==> ',len(w2v_words))
```

```
Number of unique words in Word2vec ==>  16226
```

[ ]:
```
# Words list have all the words which are in corpus
print("All the words in the courpus", len(Words))
# unique words from whole X_train
Words = set(Words)
print("The unique words in the courpus", len(Words))
# common word of w2v_words and Words
inter_words = set(w2v_words).intersection(Words)
print("The number of words that are present in both glove vectors and our␣
↪courpus", \
len(inter_words),"(",np.round(len(inter_words)/len(Words)*100,3),"%)")
```

```
All the words in the courpus 19725
The unique words in the courpus 19725
The number of words that are present in both glove vectors and our courpus 16226
( 82.261 %)
```

2.0.1 TF-IDF weighted W2V using gensim

[ ]:
```
# dictionary in which word as key and idf as value
dictionary = dict(zip(tfidf.get_feature_names(),tfidf.idf_))
```

[ ]:
```
# check for random word
dictionary['samsung']
```

[ ]:
```
8.256457189259862
```

[ ]:
```
# number of words in dictionary
len(dictionary)
```

[ ]:
```
14175
```

[ ]:
```
# get tfidf words
tfidf_words = set(tfidf.get_feature_names())
```

[63]:
```
tfidf_w2v_vectors_gensim = [] # the weighted w2v for each sentence is stored in this list
for sentence in tqdm(words): # for each sentence
    vector = np.zeros(60) # as word vectors length is 60
    tf_idf_weight =0; # num of words with a valid vector in the sentence
    for word in sentence : # for each word in a review/sentence
        if (word in w2v_words) and (word in tfidf_words):
            vec = w2v_model.wv[word] # getting the vector for each word
            # multiply idf value of word with tf of word
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence))
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_gensim.append(vector)
```

```
HBox(children=(FloatProgress(value=0.0, max=212583.0), HTML(value='')))
```

[64]:
```
len(tfidf_w2v_vectors_gensim)
```

[64]:
```
212583
```

[65]:
```
# save tfidf w2v vectors using gensim
joblib.dump(tfidf_w2v_vectors_gensim,'/content/drive/My Drive/self_case_study1/data/joblib_dumps/tfidf_w2v_vectors_gensim')
```

[65]:
```
['/content/drive/My
Drive/self_case_study1/data/joblib_dumps/tfidf_w2v_vectors_gensim']
```

[66]:
```
# total number of vectors
print('Total number of vectors = ',len(tfidf_w2v_vectors_gensim))
# length of single vector
print('Size of single vector = ',len(tfidf_w2v_vectors_gensim[0]))
```

```
Total number of vectors =  212583
Size of single vector =  60
```

Load all saved files

[67]:
```
tfidf_gensim = joblib.load('/content/drive/My Drive/self_case_study1/data/
↪joblib_dumps_glove_model/tfidf1')
w2v_model_gensim = joblib.load('/content/drive/My Drive/self_case_study1/data/
↪joblib_dumps/gensim_w2v_model')
tfidf_w2v_vectors_gensim = joblib.load('/content/drive/My Drive/
↪self_case_study1/data/joblib_dumps/tfidf_w2v_vectors_gensim')
```

[68]:
```
# w2vec words vocabulary
w2v_words = list(w2v_model_gensim.wv.vocab)
# tfidf words
tfidf_words = set(tfidf_gensim.get_feature_names())
# dictionary of tfidf and idf values
dictionary = dict(zip(tfidf_gensim.get_feature_names(),tfidf_gensim.idf_))
```

[118]:
```
text = 'how to make dictionary in python language'
```

[117]:
```
def find_similarity_gensim_trained(question,top_n):
    ''' This function will find top similar result for given query using gensim␣
↪w2v'''
    start = time.time()
    # initialize  vector for user query
    main_vec = np.zeros(60)
    # initialize tfidf weight
    weight_sum = 0
    # preprocess question
    text = preprocess_title(question)
    #splitting the sentence
    text_list = list(text.split())
    for word in text_list:
        #finding if word is present in tfidf and in w2v words
        if word in tfidf_words and word in w2v_words :
            #finding vector of word from glove model
            vect = w2v_model_gensim[word]
            #compute tfidf
            tf_idf = dictionary[word]*(text_list.count(word)/len(text_list))
            # adding vector * tfidf to main_vec
            main_vec+= (vect*tf_idf)
            # summing tfidf values
            weight_sum += tf_idf
    if weight_sum !=0:
        # devide by weight_sum
        main_vec /= weight_sum
    # find cosine similarity
    # tfidf word2vec have trained using gensim
    similarities =  cosine_similarity((main_vec).reshape(1, -1),␣
↪Y=tfidf_w2v_vectors_gensim, dense_output=True)
    # sort similarities
```

```
    sort = np.argsort(similarities[0])
    # get top similarity indices  in descending order
    similarity_index = np.array(list(reversed(sort)))
    # find top n similarities
    top_similarity_index = similarity_index[:top_n]
    # print top similarity values
    print('Top cosine similarities are␣
↪======>',similarities[0][top_similarity_index])
    similar_questions = X_train['preprocessed_title'][top_similarity_index]
    #print(similar_questions)
    similar_question_list = []
    for q in similar_questions:
        similar_question_list.append(q)
        #print(q)
    total_time = (time.time() - start)
    print('\t')
    print('Total time ===========> ',total_time)
    return similar_question_list
```

[119]:
```
top_10_similar_q = find_similarity_gensim_trained(text,10)
```

```
Top cosine similarities are ======> [0.68738857 0.68549072 0.68354058 0.67859164
0.67791326 0.67589463
 0.67470356 0.67330233 0.66850379 0.66690239]

Total time ===========>  0.5005502700805664
```

[120]:
```
print('Top 10 similar questions Using weighted TFIDF and gensim Word2Vec')
print('='*100)
print('\t')
for i,line in enumerate(top_10_similar_q):
    print('Question {} ===> {}'.format(i+1,line))
    print('='*100)
```

```
Top 10 similar questions Using weighted TFIDF and gensim Word2Vec
====================================================================================
====================

Question 1 ===> write dictionary dictionaries unknown size matrix
====================================================================================
====================
Question 2 ===> order dictionary based key daywise python
====================================================================================
====================
Question 3 ===> handling dictionary object python
====================================================================================
====================
```

```
Question 4 ===> import nested dictionary pandas yaml
================================================================================
====================
Question 5 ===> get dictionary source
================================================================================
====================
Question 6 ===> retrieve data dictionary class java
================================================================================
====================
Question 7 ===> ios dump multiple properties object dictionary using predicate
kvc
================================================================================
====================
Question 8 ===> iterate list populated nested dictionary python
================================================================================
====================
Question 9 ===> getting json dictionary list values python
================================================================================
====================
Question 10 ===> append lines gz file dictionary python
================================================================================
====================
```

3 Model-2 Doc2Vec

Reference : https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html#sphx-glr-auto-examples-tutorials-run-doc2vec-lee-py

There are two implementations:

1. Paragraph Vector - Distributed Memory (PV-DM)

2. Paragraph Vector - Distributed Bag of Words (PV-DBOW)

[86]:
```
def read_corpus(filename, tokens_only=False):
    for i, line in tqdm(enumerate(filename)):
        tokens = gensim.utils.simple_preprocess(line)
        if tokens_only:
            yield tokens
        else:
            # For training data, add tags
            yield gensim.models.doc2vec.TaggedDocument(tokens, [i])
```

[87]:
```
# train corpus
train_corpus = list(read_corpus(X_train['question_content']))
```

```
HBox(children=(FloatProgress(value=1.0, bar_style='info', max=1.0), HTML(value='')))
```

[88]:
```
# check for first 5 questions
train_corpus[:5]
```

[88]:
```
[TaggedDocument(words=['react', 'navigation', 'stack', 'navigator', 'using',
'react', 'navigation', 'app', 'tab', 'navigator', 'contains', 'stack',
'navigator', 'want', 'use', 'stack', 'navigator', 'user', 'top', 'screen',
'stack', 'nav', 'going', 'back', 'stack', 'want', 'return', 'false', 'way',
'works', 'always', 'true', 'unless', 'first', 'tab', 'screen', 'app', 'starts',
'takes', 'different', 'tabs', 'solution', 'work', 'around', 'help', 'highly',
'appreciated'], tags=[0]),
 TaggedDocument(words=['display', 'field', 'edit', 'form', 'currently', 'try',
'place', 'field', 'edit', 'form', 'field', 'display', 'errors', 'console',
'terminal', 'wont', 'example', 'display', 'either', 'page', 'load', 'simply',
'blank', 'way', 'use', 'fields', 'edit', 'form'], tags=[1]),
 TaggedDocument(words=['keep', 'border', 'trying', 'set', 'flutter', 'focus',
'turns', 'however', 'would', 'like', 'keep', 'border', 'titles', 'whenever',
'focused', 'expect', 'color', 'change', 'border', 'titles', 'write'], tags=[2]),
 TaggedDocument(words=['load', 'image', 'work', 'emulator', 'trying', 'get',
'image', 'load', 'onto', 'app', 'imageview', 'however', 'whenever', 'try',
'running', 'tap', 'image', 'want', 'load', 'says', 'unfortunately', 'stopped',
'try', 'app', 'emulator', 'nexus', 'running', 'works', 'perfectly', 'fine',
'wondering', 'done', 'wrong', 'make', 'work', 'thanks', 'code', 'xml', 'file'],
tags=[3]),
 TaggedDocument(words=['stop', 'running', 'method', 'want', 'stop', 'currently',
'running', 'method', 'short', 'duration', 'without', 'using', 'thread'],
tags=[4])]
```

[89]:
```
X_train.shape
```

[89]:
```
(212583, 9)
```

3.0.1 Training the model

[104]:
```
# define model
model = gensim.models.doc2vec.Doc2Vec(vector_size=100, min_count=10,␣
 ↪epochs=60,workers=-1)
```

[105]:
```
# build vocab
model.build_vocab(train_corpus)
#Essentially, the vocabulary is a dictionary (accessible via model.wv.vocab) of␣
 ↪all of the unique words extracted
#from the training corpus along with the count (e.g., model.wv.vocab['penalty'].
 ↪count for counts for the word penalty).
```

[106]:
```
# count of word python in vocab
print(model.wv.vocab['python'])
```

```
Vocab(count:15044, index:71, sample_int:4294967296)
```

[107]:
```
print(model.corpus_count)
```

```
212583
```

[108]:
```
# train model
model.train(train_corpus, total_examples=model.corpus_count, epochs=model.
 ↪epochs)
```

[111]:
```
# save model
joblib.dump(model,'/content/drive/My Drive/self_case_study1/data/doc2vec/
 ↪doc2vecmodel')
```

[111]:
```
['/content/drive/My Drive/self_case_study1/data/doc2vec/doc2vecmodel']
```

[ ]:
```
# load model
model = joblib.load('/content/drive/My Drive/self_case_study1/data/doc2vec/
 ↪doc2vecmodel')
```

[109]:
```
# shape of the vector
model.docvecs[0].shape
```

[109]:
```
(100,)
```

[115]:
```
test_corpus = list(read_corpus('how to make dictionary in python',␣
 ↪tokens_only=True))
vector = model.infer_vector(test_corpus[0])
simi = (model.docvecs.most_similar([vector], topn=10))#this algorithm by␣
 ↪default calculates cosine similarity as a distance metric
print(simi)
```

```
HBox(children=(FloatProgress(value=1.0, bar_style='info', max=1.0), HTML(value='')))


[(183610, 0.43154197931289673), (196860, 0.4177968204021454), (127069,
0.40544310212135315), (176202, 0.4016878306865692), (181276,
0.39809274673461914), (129982, 0.3949902653694153), (132343,
0.3900105357170105), (10880, 0.385919451713562), (184499, 0.3844781219959259),
(113944, 0.38275274634361267)]
```

[116]:
```
sim_doc2vec = []
for i in range(len(simi)):
  idx = (simi[i][0])
```

```
  se = X_train["question_content"][idx]
  print(se)
  print('='*100)
  #sim_doc2vec.append(se)
```

```
multiple parameters post mvc ok two elements page located single form need
posted back controller via jquery ajax method problem parameters correctly post
set strings include parameter name value get irritating way make sure mvc map
parameters correctly take correct values really want jquery call actual code
using serialize duh
====================================================================================
====================
center menu items responsive navbar schools nice example nav bar respond smaller
screen https www schools com howto tryit asp filename js menu items centered
instead aligned left setting makes stack vertically
====================================================================================
====================
pandas write dataframe parquet format append trying write file format introduced
recent pandas version mode however instead appending existing file file
overwritten new data missing write syntax read syntax
====================================================================================
====================
load videojs ajax trying load ajax video working videojs manage make work ajax
load reading lot could find solution show example check link working video
trying load previous link ajax load code run ajax call check also links entered
know videojs initialized fine tried lot alternatives without success help much
appreciated thank
====================================================================================
====================
prepopulate form struts database values want show edit form user change old data
want form values picked please tell struts
====================================================================================
====================
jqgrid perform search locally entire data set without considering pagination
jqgrid search functionality perform search locally entire data set without
considering pagination nearly records database jqgrid pagination search
something perform search entire dataset irrespective page grid request database
everytime loaded thanks advance
====================================================================================
====================
rails activerecord clause accepts dynamic parameter ruby web define parameters
method activerecord defined accepts dynamic parameters went though blogs
websites could find useful resources
====================================================================================
====================
loading page jquery ui tabs page div tag via ajax parent page call jsp page div
tag want load div tag page jsp already jquery ui tabs contents means want page
```

tab loaded parent page without writing code jquery tabs call jsp
==========
==========
got using unsupported command line flag disable web security stability security suffer error got error using selenium rc google chrome using unsupported command line flag disable web security stability security suffer know whats issue chrome
==========
==========
regexp matching protocol relative url beginning string writing short dynamic script piece makes sure url starts either http https http https merely technique regexp far could give would work know nearly get say http https slash slash beginning string whatever comes urls
==========
==========