**Comparing Dates**

Author: Nick Becker (nbecker@mitre.org)

Last updated: 10 October 2017

*Note: The software described by this document is a prototype and is therefore subject to change at any time without updates to this document.*

[Goal](#goal)

[Algorithm considerations](#algorithm_considerations)

[Summary of algorithm](#summary_of_algorithm)

[Algorithm details](#algorithm_details)

[An example](#an_example)

*Goal*

Given two dates, the goal was to come up with a means of producing a score between 0 and 1 (inclusive) that indicates how similar the two dates are to each other, no matter how fully-specified either date is.

*Algorithm considerations*

1. One of the biggest complicating factors in designing an algorithm for computing date similarity is that there are at least two ways to think of date similarity: in terms of temporal proximity or in terms of string similarity. To help illustrate this distinction, let us consider the following pair of dates (dates are in YYYY-MM-DD format):

A: 1969-01-31

1970-01-31

B: 1969-01-31

1996-01-31

In terms of temporal proximity, the dates in A should yield a similarity score higher than those in B: the dates are a single year apart in A and 27 years apart in B.

However, if we are interested instead in string similarity, the dates in B should yield a similarity score higher than those in A: the strings representing the dates in B are separated by a single transposition of characters, whereas the strings representing the dates in A are separated by two character substitutions.

1. Another factor that complicates the comparison of dates is that they can assume a number of different textual forms exist in different calendar systems. For example, all of the following refer to the same day.

1969-01-31

01-31-1969

1969/01/31

01/31/1969

31 Jan 1969

31 January 1969

January 31, 1969

Jan. 31, 1969

1388-11-13

1388/11/13

13 Dhu al-Qi'dah 1388

13 Dhul-Qa`dah 1388

Dhu al-Qi'dah 13, 1388

Dhul-Qa`dah 13, 1388

Note that not only does the format of the date vary (YYYY-MM-DD, MM-DD-YYYY, YYYY/MM/DD, etc.), but dates in different calendar systems can appear to be drastically different in terms of month names and years.

*Summary of algorithm*

After extracting as much information about the provided dates as possible, including the calendar system, the dates are passed through five different sub-scoring systems which compare the dates along lines of temporal proximity and string similarity. The final score is a weighted average of these sub-scores.

*Algorithm details*

Note: The following describes an algorithm employed by prototype software. Precise details and numerical values are subject to change.

1. Both dates are parsed; year, month, day, and calendar system are extracted and stored.
2. The dates are then passed through five different sub-scoring systems:
   1. Year-month-day score: provides a baseline similarity score. Year-to-year, month-to-month, and day-to-day comparisons are made. Each match contributes a pre-specified award whose value depends on the date component being compared (i.e. year, month, or day). These different award values allow the algorithm to reflect the difference in how important each component is in determining overall date similarity. The final score is a sum of all contributed awards.
   2. Levenshtein similarity score: captures string similarity. Both dates are converted to simple string representations of the form “YYYYMMDD,” and a Levenshtein-distance-based similarity metric is computed.
   3. Proximity score: awards dates that are temporally near each other. A score is assigned based on how close the two dates are to each other, to best approximation. This value is calculated as 1 – (distance in days / 1825), where 1825 refers to the number of days in five Gregorian years. The final score is proportional to the proximity (i.e. closer dates receive a higher score).
   4. Cosine similarity score: captures fine-grained temporal differences. Both dates are converted to numerical vectors (approximate numerical vectors if either date is underspecified), and the cosine similarity of the two vectors is computed. This score is very fine-grained: the difference between the cosine similarity of two very different dates and the cosine similarity of two very similar dates is quite small. Thus, this measure primarily serves as a means of making the overall measures “more continuous” within the range 0.0-1.0, thus reducing the likelihood that two date comparisons will yield the same final similarity.
   5. Specification score: awards comparisons where both dates are fully specified. If both dates are fully specified, the final score is 1; else the score is 0.
3. Once all four sub-scores are computed, their weighed average is computed to arrive at the final score. The weighting of each component ideally reflects the priorities of the application’s domain but is configured to yield reasonable results by default.

*An example*

Let us consider a concrete example: comparing “1969-01-31” and the underspecified “January 1970”.

1. Parse each date:

1969-01-31: year = 1969

month = 1

day = 31

January 1970: year = 1970

month = 1

day = 00

2.a. Year-month-day score. In this example, we will use the following year, month, and day awards:

Year 0.40

Month 0.35

Day 0.25

Using these values as we make the year-to-year, month-to-month, and day-to-day comparisons, we end up with the following year-month-day score:

Comparison Award

1969-1970 0.00

* 1. 0.35

31-00 0.00

Sum 0.35

Thus, our final year-month-day score is 0.35.

2.b. Levenshtein similarity score. The dates are first converted to strings:

1969-01-31: “19690131”

January 1970: “19700100”

The Levenshtein similarity score is defined as 1 – (Levenshtein distance / length of longer string), so the score for these strings is 1 – (4 / 8) = 0.5.

2.c. Proximity score. Because one of the dates we are comparing is underspecified, the best proximity score that can be computed is an estimate, derived by setting the underspecified date to “close” fully specified date:

1970-01-00 (underspecified original) 🡪 1970-01-16 (fully specified approximation)

Temporal distance between dates 1969-01-31 and 1970-01-16: 350 days

Plugging this value into the equation provided above, we get 1 – (350 / 1825) = 0.80822.

2.d. Cosine similarity score. As with the last step, for computing cosine similarity, we first must approximate an exact date for each underspecified date that we are considering. We will use the same fully specified approximation that was used above.

Next, each date is vectorized, and the cosine similarity of the two vectors is computed. For the sake of simplicity, the details of this calculation are omitted:

cosSim([1969, 01, 31], [1970, 01, 16]) = 0.999981847771675…

2.e. Specification score. One of the dates here is underspecified, so the specification score for this pair is 0.

3. Once each sub-score has been computed, we then compute their weighted average. For this example, we will use the following weights:

Metric Weight

YMD score 0.25

Levenshtein similarity score 0.25

Proximity score 0.30

Cosine similarity score 0.15

Specification score 0.05

These weights result in the following final score calculation:

Final similarity score = 0.25(0.35) + 0.25(0.5) + 0.30(0.80822) + 0.15(0.9999…) + 0.05(0)

Final similarity score = 0.60496

Thus, using this algorithm, the similarity of “1969-01-31” and “January 1970” is 0.60496.