Using Alignment Algorithm for Location Prediction on Google Location Data

Olga Groh*, Johann Götz†, Fabian Frölich‡
Faculty of Electrical Engineering and Computer Science
University of Kassel,
Kassel, Germany
Email: {*o_groh, †uk017305, ‡f.froelich}@student.uni-kassel.de

***Abstract*—Nowadays, with the position history of a person, it's possible to predict the location where the person will be next. This approach is interesting in a wider field of application, such as rescue service, crime detection, advertising or personal smart-home systems. In this paper we try to predict a persons location with the Smith-Waterman alignment algorithm, which is commonly used in bioinformatics to compare segments of protein sequences. As data set we used Google Location History from different mobile devices. To minimize redundancy in the sequences, we introduced a compression method. The Smith-Waterman algorithm has the potential to be a decent prediction algorithm.**

***Index Terms*—location prediction; alignment algorithm**

I. Introduction

The knowledge about a person's location plays a very important role in today's digital world. An exact location in all imaginable areas can be helpful or even save lives. Another step forward is the prediction of the future location of a person. With that, one can not only acquire the information of the current location of said person and also determine with high probability where he/she will be located next. To further illustrate the importance of this technology, some likely scenarios will be presented in the following.

One life saving possibility using this technology, presents itself in relation to the elderly. Provided with the right devices, it is possible today to determine the state of health of a person and take action accordingly. Should this state take a turn for the worse or when the person has taken a fall, it can be possible to get the required help in time. In those situations the location of a person is of great importance. With the location ascertainment it will be possible to find the person with the correct latest location information. Furthermore, it will be possible using location ascertainment to predict the location even if the device is is left behind or if it malfunctions.

An exact location is helpful in other scenarios relating to danger and rescue as well. If one imagines an accident on the freeway, it is very important for the rescue to know the exact road on which the accident took place. In this case the exact position is more important than the prediction. However, there are several situations, where the possibility of location prediction plays a more important role. Some examples of that would be rescuing hikers stuck on the trail due to bad weather, or helping people after a natural disaster.

Additional importance for location ascertainment of people is in the field of crime fighting and -solving. It could also be profitable for companies because the use of this technology allows them to project and distribute personalized marketing in predetermined locations. But even a private user would find benefits in the determination of his/her location, when one bears in mind the constant rise in use of smart-home-technology. Using the location information of the user, the home system knows when he/she will be home and even the right time to turn on the heating.

All the scenarios described above are dependent on determining and predicting the location of a person. It is also not problematic to acquire the data needed for this in today's digital world. In the last ten years the smartphone has proven itself to be a useful gadget and a permanent companion. But the power of this device is still underrated by most. With the gathered and in most cases also saved data, it is possible to make assumptions about the owner of the phone. Even more, it is possible to create a very detailed personal profile. With the gathered location data, using the right algorithms and methods, a position prediction becomes possible.

In the following, an attempt will be made to create a location prediction using alignment. The used positioning data was gathered in advance from various people and smartphones. Using this information and an alignment algorithm, an effort will be made to predict the next location.

II. Related Work

Plenty of research material exists for location prediction. Alignment algorithms offer a similarity measure for arbitrary continuations of data points which are also called sequences. In other words the use for such algorithms allows us to check and compare the similarity of two sequences. Common adaptations often take place in bioinformatics. Here, DNA (Deoxyribonucleic acid) and protein sequences are being analyzed to determine evolution based relations. An example for this purpose can be found in [11].

Sigg et al. [9] describe an approach where context sequences can be predicted by using alignment. The utilization of alignment algorithms allows Sigg et al. to find the most similar partial sequences and with the aid of these to predict the next entry. Among other things they use the Needleman-Wunsch algorithm [7] for that purpose which finds a global optimum

by using the backtracking algorithm. A similar approach is the Hirschberg algorithm [5] which is also a global algorithm similar to Needleman-Wunsch.

Our attempt is based on alignment as well. Unlike [9] we make use of the Smith-Waterman algorithm [10] within our location prediction approach. In contrast to the Needleman-Wunsch and Hirschberg algorithm, this algorithm is able to find a local optimum. One benefit would be in fact, that local analogy can't be missed which is often the case with global consideration.

Apart from the use of alignment algorithms there are two additional approaches for location prediction. Craig et al. [3] use a clustering approach for defining whereabouts of one person trough clusters. This is done by dividing a location in preferably small regions and then combining similar adjacent regions to bigger clusters. Based on these regions the next objective is to predict the next location where a person will be situated. On the basis of the current or previous location the algorithm checks which route is the user is currently on. The cluster, which contains the user's next possible goal, is then determined with the aid of this information. Unlike our approach, clustering doesn't provide an optimal solution but rather it performs a reduction of data and thus enables the use for it on under-performing devices.

Rjeily et al. [8] use a sequence prediction approach to detect heart deficiencies of patients. For this to happen, discrete and continuous data sequences, for example, are applied to determine the chest pain index from 1 to 4 and high blood pressure values. Therefore their approach is based on the compact prediction tree algorithm extension [4]. This algorithm is capable of estimating the next symbol of a given sequence which places the patients in a certain risk group. In doing so the sequence is used to check the ranking of the patient in the risk category. In comparison to this presented approach the algorithm of Rjeily et al. operates on complex high-dimensional data hence it is too sophisticated for our data. Because the data is composed of pure discrete symbols, the application of such an algorithm would be unsuitable for our approach.

All in all the sequence prediction has several uses and potentially qualifies for location prediction.

III. Conception

Nowadays, almost anybody is using a mobile device with a GPS (Global Positioning System) on-board.

The benefit of it is that Google Location History data [6] is gathered automatically by every mobile device that uses Google services. The daily logged data contain accurate position coordinates and timestamps right up to location names and inaccuracies.

The data originates from the Google Location project [2] and is available as a Comma-separated-values-file (CSV-file) that is represented in extracts in the following Table I.

For location prediction individual datasets of the CSV-file are sorted by the `timestamp`. The `location` describes the location name of the user while `logitude`, `latitude` and `altitude` represent the coordinates. Additionally, the `accuracy` specifies the location precision. For our approach we need merely location names in a correct and chronological order.

TABLE I
Example of collected Google Location Data

timestamp	accuaracy	longitude	latitude	altitude	location
1300040100	30	9.500.700	51.330.700	0	Home
1300039200	31	9.500.700	51.330.700	0	Home
1300028400	31	9.500.300	51.330.300	0	Sport
1300024800	31	9.500.700	51.330.700	0	Home
1300022400	31	9.500.800	51.330.800	0	Friend
1300021200	32	9.500.800	51.330.800	0	Friend
1300017600	32	9.500.700	51.330.700	0	Home
1300010400	31	9.500.500	51.330.500	0	Work
1300003200	30	9.500.500	51.330.500	0	Work
1299998200	31	9.500.700	51.330.700	0	Home

The idea now is to predict the next location of a person with the previous location data. Now we consider the data as a long sequence like depicted in Fig. 1. Then a search sequence with the last x characters is extracted from it. Finally the alignment is applied to the entire sequence to find the most similar local sequence. All following characters form the prediction sequence of length l.

The algorithm that is suited best for this kind of problem is the Smith-Waterman algorithm because it can be utilized for sequences with different lengths to find the smallest possible match locally. Another benefit is that the sequences mustn't be identical so that even similar sequences can be found as well. Algorithms like Needleman-Wunsch and Hirschberg are not suitable for it because those can only calculate the biggest, most similar sequence globally.

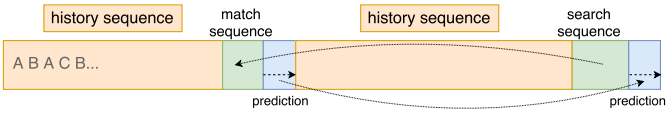

Fig. 1. Concept of sequence alignment and prediction

Smith-Waterman algorithm is a local optimization algorithm which operates on two character string sequences. First, both sequences are arranged in a way that they form a matrix so that its columns and rows are initialized with zeros. Beginning with the first index in the matrix, every character from `sequence A` is compared iteratively to every character from `sequence B`. Then, with the help of recurrence values, a similarity score is calculated. If two characters match, then the score is increased by a fixed recurrence value. The score decreases accordingly if the characters don't match. Likewise it can happen that the sequences are indeed similar but by a gap no longer identical. During a gap situation in a sequence it is either the case of insertion or deletion. The next entries are equally calculated dependent on the previous value. Unlike in the Needleman-Wunsch algorithm, the matrix entry in the Smith-Waterman algorithm cannot be negative. However, if the entry falls into the negative section, its value is then set to zero. After that the algorithm looks for the highest value in the

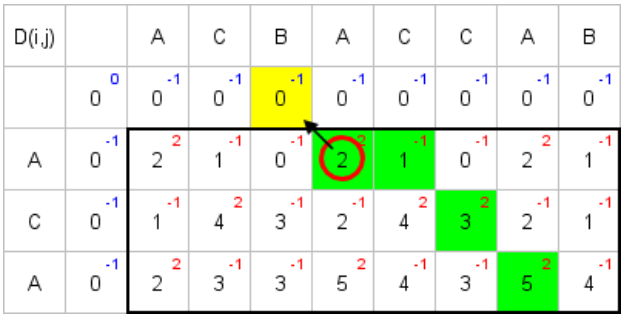

Fig. 2. Filled-in Smith-Waterman matrix with traceback [1]

entire matrix. This is the optimal value. Alignment is optimal if and only if it contains such a value. A matrix can posses multiple maxima therefore the alignment can be optimal in several spots.

There are multiple criteria for choosing recurrence values. If the penalty for gaps is higher, the algorithm finds more mismatches in the sequence. On the other hand, if the penalty for mismatches is higher, it is more likely that the algorithm finds more gaps in the sequence. To get the balance between those possibilities, default recurrence values can be used: *match=2*, *mismatch=-1* and *gap=-1*.

Next, the respective most similar characters are ascertained by the traceback algorithm. Based on the maximum value the path is then traced back diagonally to the north west, exclusively up until the value zero. At last, the indices result from the alignment of start and end positions of the match sequence.

In addition, the matrix can contain more than one of the highest scores. In this case our program takes the chronologically first result. This is important because the possibility to get a longer prediction sequence is much higher. If the result is located mainly at the end of the sequence, the prediction algorithm has much less characters to work with.

A. Example

To illustrate the use of the algorithm we consider two sequences `ACBACCAB` and `ACA` to be aligned, like shown in Fig. 2.

1) The first column and row of the matrix should be initialized with zeros.
2) Beginning with the first empty slot in the upper left corner the score is calculated iteratively with the help of recurrences. The score for `(A,A)` results in the value 2, because there is a match, whereas for `(A,C)` it results in 1, because the characters don't match (so the recurrence is `gap=-1`) and because the previous value is 2, the recurrence is subtracted.
3) If the matrix is calculated completely, the highest value, in this case it is 5, is found.
4) Now, the traceback algorithm searches for the right path backwards diagonally to the north west until the value 0. As of now, the path has the length of 5. As said before, the calculated value 0 does not count. That means the path length is 4.
5) By reading from the matrix the result for the local alignment is `A-CA`, because the algorithm found a gap.

IV. Implementation

In Chapter III we already explained, that we require for the alignment only timestamps and descriptions from the data.

The data from the CSV-file is separated by semicolons from each other. Descriptions are extracted from the lines and then sorted by the timestamps. During the row-wise reading each location name is assigned to a different character from the alphabet. This is necessary because the implementation of the algorithm expects one particular character for the calculation. Adjustments and Optimization on strings for location name are also viable. To avoid duplicates, the first character from the location name is not used but rather a beforehand defined character. This character string is sorted by timestamps and it forms a so-called *history sequence*.

The implementation uses two types of depiction of the history sequence - the entire character string of location names and the summarized. In the first depiction all available, recorded data in the history sequence is carried over. The summarized depiction summarizes all data entries based on time intervals. When each character is imported, the algorithm checks whether the sequent characters can be summarized. Here, the chosen time interval is 30 minutes. Within this time interval, if the character doesn't change it will be ignored. If another character (this means a location change) follows after a considered one then this character is included in a compressed sequence. Is the subsequent character equal but the time interval is exceeded, the character remains in the sequence. Here, a long-term stay in one location is pictured. By applying the compression we obtain a clearly shorter sequence which is presented in-depth in the following Table II.

TABLE II
Comparison uncompressed and compressed history sequence

Data set	uncompressed	compressed	reduction
Sequence 1	1858	436	76,53%
Sequence 2	5499	1015	81,54%
Sequence 3	2222	581	73,85%

The next step will be to define *x* characters at the end of the history sequence. Now, both sequences are handed over to the Smith-Waterman algorithm in order to compute the score matrix. In our program, the algorithm operates on the default recurrences values. A matrix cell with the highest score and with the indices of the found local sequence is expected as output so that we can get the values of the history sequence directly. If the matrix contains more than one highest score, the first found one is taken. All the characters with the selected length *l* can now be read off the history sequence. We finally obtain the predicted sequence.

$$Home \rightarrow A, Work \rightarrow B, Friend \rightarrow C, Sport \rightarrow D$$

Fig. 3. Example of location data mapping

A. Example

Let us assume that the extract from a CSV-file consists of unsorted data like shown in Table I.

1) First, the data sets are sorted by a timestamps and reduced to two columns, like presented in Table III.
2) Each location name now represents an individual character from the alphabet. This is shown in Fig. 3.
3) When every character is arranged into a character string, it represents the sequence `ABBACCADAA`.
4) Normally, the sequence is much longer but to demonstrate the compression, we remove duplicate characters in order to receive a shortened sequence `ABBACADA`. Since the time interval between the first and the second `B` is greater than 30 minutes, they won't be shortened but the rest is. We now have gotten our history sequence.
5) Since also the history sequence is very short, we say, that our search sequence consists only of the character `A` which is found at the end.
6) After applying the Smith-Waterman algorithm , we receive `B` as the predicted next character of the sequence because it follows after the first found character `A` in the history sequence. Again, the algorithm operates best on much longer sequences.

TABLE III
SORTED, USABLE GOOGLE LOCATION HISTORY DATA

timestamps	location
1299998200	Home
1300003200	Work
1300010400	Work
1300017600	Home
1300021200	Friend
1300022400	Friend
1300024800	Home
1300028400	Sport
1300039200	Home
1300040100	Home

V. EVALUATION

To evaluate the accuracy of the prediction, we try to make predictions on existent data to compare it to available data sets. The used data sets are comprises time periods from one to several months. Out of the whole sequence, we use a certain amount of data to be our test sequence. Hence the rest becomes the history sequence from which we extract the search sequence like before. We define all the characters after the search sequence (which are now the first few characters of the test sequence) as the *expected prediction*. Now, we apply the Smith-Waterman algorithm to the search and history sequence to obtain a *possible prediction*. If a prediction exists, we compare it to the expected prediction to verify the correctness of the process. This evaluation method is depicted in Fig. 4.

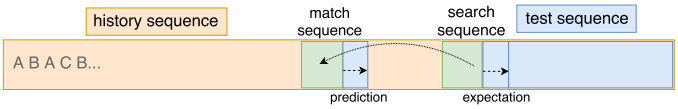

Fig. 4. Concept of prediction sequence testing

Furthermore, we consider the impact of alteration of the search sequence length k and the prediction sequence length l for uncompressed (Fig. 5) and compressed (Fig. 6) sequences. For both sequences, we observe the first three predicted characters with different search sequence lengths. For the accuracy, we shift the search sequence of all data sets by a certain amount to obtain multiple results.

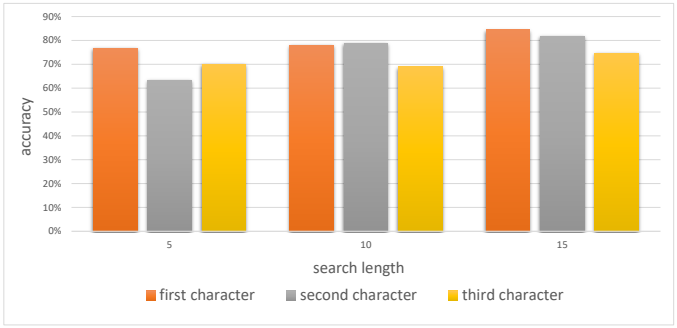

Fig. 5. Predicted values from original sequence

As one can see, the accuracy of an uncompressed sequence rises with the search sequence length. For the 15 character search length we reached 84.44% accuracy for the first predicted character which was the best result. The accuracy for five and ten character search length were very similar. They reside at 76.67% and 77.78%. Additionally, the accuracy for each of the three characters stays about the same within a certain range but improves overall because of the fact that the uncompressed sequence generally has multiple repetitions of the same characters.

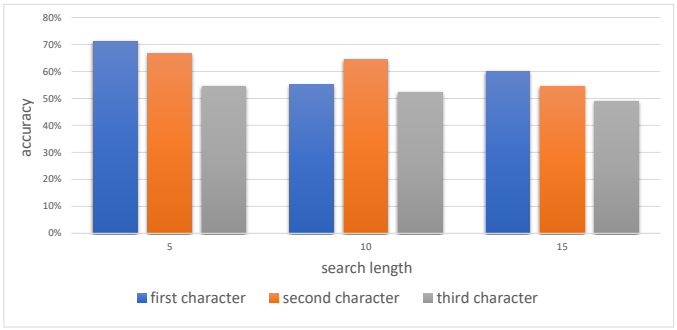

Fig. 6. Predicted values from the compressed 30 minutes sequence

Meanwhile, the values for the compressed sequence aggravate in terms of accuracy when the search sequence length is increased. The best result for the five character search

length accuracy is 71.11%. For ten character search length the accuracy is 55% and for 15 character search length it is 60%. However multiple advantages are given when using this method. Through compressing the sequence, the redundancy of pointless character repetitions is avoided so the actual prediction accuracy of useful characters is overall improved. Though it may seem, that the accuracy for uncompressed sequences is a little higher than for the compressed one, we should consider that it does not tell the real usefulness of the prediction because of random time intervals. So even if the accuracy for compressed sequences is lower, it is much more precise in that way and also faster regarding the sequence length.

VI. Conclusion

The ability to predict a person's next location is beneficial to many applications. An example for such kind of application can be found in a medical environment. In this case, location prediction could be particularly helpful in detecting accidents of elderly persons. If an older person's behavior significantly differs from the prediction on several occasions, it could indicate that the person needs help.

The presented approach collected Google Location History data from several users. Based on this, our approach was to predict the next location of a user by applying alignment techniques. Therefore, we converted the data into a sequence and with the Smith-Waterman algorithm we tried to determine a possible continuation of the sequence. Summarizing the evaluation results it can be said that using the compressed sequences lowers the prediction accuracy. On the other hand this data reduction significantly reduces the needed storage space and data redundancy which could be beneficial for the use in mobile devices. Therefore, we suggest to use there original data on high-performance devices and the compressed sequences in low-performance devices.

In our future work, we want to improve the performance of our approach in order to use it in real-time applications. In order to improve the algorithm and find the best match sequence out of multiple possible matches with the highest score, we could compare each of their predictions to take one with the best accuracy. Additionally we want to take more data into account. For example, the exact coordinates provide additional information to the locations given by Google Location History and could enhance the prediction. Furthermore, we want to extract the data directly from Google Location History. One possible application is the use of our approach in the Smart Home environment. Based on the user's predicted location the heater or coffee machine could be automatically turned on at a certain time.

References

[1] "Basic-algorithms-of-bioinformatics applet," http://baba.sourceforge.net/, accessed: 2018-03-22.

[2] "Google maps - timeline," https://www.google.com/maps/timeline, accessed: 2018-02-15.

[3] I. Craig and M. Whitty, "Region formation for efficient offline location prediction," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 66–73, 2017.

[4] T. Gueniche, P. Fournier-Viger, and V. S. Tseng, "Compact prediction tree: A lossless model for accurate sequence prediction," in *International Conference on Advanced Data Mining and Applications*. Springer, 2013, pp. 177–188.

[5] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences," *Communications of the ACM*, vol. 18, no. 6, pp. 341–343, 1975.

[6] A. Kirmse, T. Udeshi, P. Bellver, and J. Shuma, "Extracting patterns from location history," in *ACM SIGSPATIAL GIS 2011*, http://www.sigspatial.org/, 2011, pp. 397–400.

[7] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[8] C. B. Rjeily, G. Badr, A. H. A. Hassani, and E. Andres, "Predicting heart failure class using a sequence prediction algorithm," in *2017 Fourth International Conference on Advances in Biomedical Engineering (ICABME)*, Oct 2017, pp. 1–4.

[9] S. Sigg, S. Haseloff, and K. David, "An alignment approach for context prediction tasks in ubicomp environments," *IEEE Pervasive Computing*, vol. 9, no. 4, pp. 90–97, 2010.

[10] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[11] M. S. Waterman, *Introduction to computational biology: maps, sequences and genomes*. CRC Press, 1995.