T^3: On Mapping Text To Time Series

Tao Yang and Dongwon Lee[⋆]

The Pennsylvania State University, University Park PA 16802, USA

Abstract. We investigate if the mapping between text and time series data is feasible such that relevant data mining problems in text can find their counterparts in time series (and vice versa). As a preliminary work, we present the $\mathbf{T}^3$ (*T*ext *T*o *T*ime series) framework that utilizes different combinations of granularity (e.g., character or word level) and n-grams (e.g., unigram or bigram). To assign appropriate numeric values to each character, T^3 adopts different space-filling curves (e.g., linear, Hilbert, Z orders) based on the keyboard layout. When we applied T^3 approach to the "record linkage" problem, despite the lossy transformation, T^3 achieved comparable accuracy with considerable speed-up.

1 Introduction

Despite significant advancement in each area, data mining research in textual data (e.g., web pages of search engines, citations of digital libraries, relationship data in social networks) and time series data (e.g., network traffic observations, daily fluctuations of stock prices) have not been developed in a close synchronization. New techniques developed in one area do *not* easily get carried over to the other area. This is partly due to the fact that although both deal with many similar problems such as defining appropriate distance functions or finding interesting patterns, their subject domains are different – i.e., alphabetical strings vs. numerical signals. Therefore, toward this lack of connection between the emerging time series and the traditional text mining approaches, in this paper, we investigate if there exists feasible transformation between two data types such that relevant data mining problems in one data type can find their counterparts in the other type. We are interested in whether and to what extent the performance of mining solutions developed in one domain can be improved over the solutions in the other domain.

Toward this objective, as a preliminary work, we present the $\mathbf{T}^3$ (*T*ext *T*o *T*ime series) framework to map text data to time series data. During the transformation of the entire text corpus, T^3 utilizes different combinations of granularity (i.e., character level or word level) to extract text units from strings. Furthermore, T^3 utilizes n-grams (i.e., unigram, bigram or trigram) to form subsequences of text units. In order to assign appropriate numeric values to each character, T^3 adopts different space-filling curves (i.e., linear, Hilbert, Z orders) based on the keyboard layout. In addition, to associate real values to each token, T^3 uses the tf-idf weight of the traditional weighting scheme from information

[⋆] Partially supported by IBM and Microsoft gifts.

retrieval and text mining. We apply the T^3 framework to the *Record Linkage* problem, one of the traditional data mining problems, to determine whether or not two entities represented as relational records are approximately the same. Through extensive experiments using both real and synthetic data sets, the efficacy of our proposed schemes is experimentally validated. To the best of our knowledge, this is one of the first attempts to solve a text mining problem in time series domain. Our experiments reveal that T^3 shows comparable accuracy (despite the *lossy* transformation in T^3) when compared to a popular distance measure (e.g., Levenshtein distance) in text domain. However, T^3 also achieves much improved speed-up thanks to the numerical data of time series domain.

2 Related Work

Time series data mining has received tremendous attention in the data mining community during the last decade. Many time series representation methods such as Discrete Fourier Transformation (DFT) [7], Discrete Wavelet Transformation (DWT) [3], Piecewise Aggregate Approximation (PAA) [12], Singular Value Decomposition (SVD) [7] and Symbolic Aggregate approXimation (SAX) [13] etc. have been proposed together with the corresponding similarity measures such as Euclidean Distance (ED) [7], Dynamic Time Warping (DTW) [1], Distance based on Longest Common subsequence (LCSS) [17] and so on. Recently, [5] summarized and evaluated the state-of-the-art representation methods and similarity measures for time series data through extensive experiments.

On the other hand, the general linkage problem has been known as record linkage [2, 8], merge-purge [10], citation matching [14], object matching [4], entity resolution [16], authority control [18], and approximate string join [4, 9], to name a few. Excellent survey papers [6, 19] provide the latest advancement of the linkage problem. In our recent work, we presented the novel idea of solving the record linkage problem using BLAST, one of the most popular gene sequence alignment algorithms in Bioinformatics [11] . We proposed four variations of linkage solutions to translate text data into DNA sequences and demonstrated the good combination of accuracy and speed of applying BLAST to record linkage. However, none of these existing works attempted to solve the linkage problem using time series mining techniques as we did in this paper. Recently the authors in [15] mentioned a

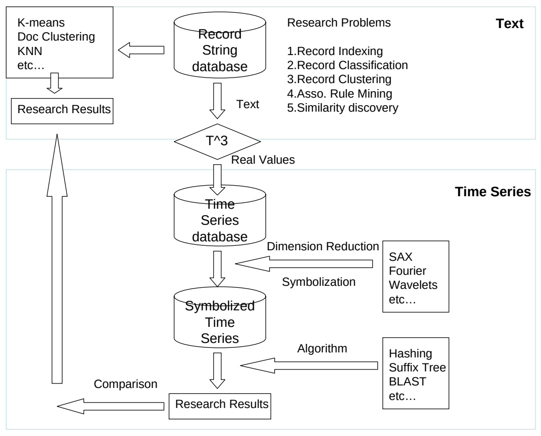


Fig. 1. Overview of the T^3 framework.

method to transform text into a time series representation in the case of translating biblical text in both English and Spanish. The basic idea is to convert the bible text into bit streams based on the occurrences of a particular word in the text. Then a time series is generated based on the number of word occurrences within a predefined sliding window across the bit streams. Although it is useful in the case of generating time series for the translation versions of the same text in two different languages, their method can not been directly applied to the record linkage problem because each record may have different sets of words and it would be hard, if not impossible, to find a common word among them before the time series conversion. To the best of our knowledge, our effort is one of the first attempts to solve the record linkage problem in time series domain.

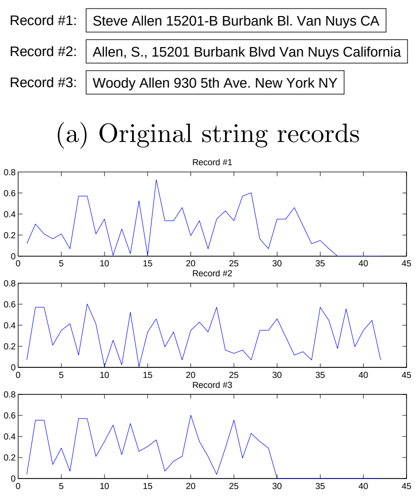
3 The T^3 Framework

The basic idea of T^3 is illustrated in Figure 1. Instead of solving data mining problems on string/text data, we first scan the string database using the proposed transformation schemes in T^3. After the string database is mapped to a new time series database, we then employ dimension reduction and symbolization techniques directly on the real values of time series. In general, T^3 serves as a convenient bridge to connect two subject domains: numerical signals and alphabetical strings. Therefore, our approach can be considered as a novel complement to existing text mining algorithms which were solely built for generic use based on string manipulation. We illustrate our idea using a simple example in Figure 2. The first two records are referring to the same person and the third record belongs to a different person. We can easily see that the time series of the first two records preserve similar shapes in real-value domain (with some shifting) while the time series of the third record has a rather different shape. Given any sequence s from a database of textual sequences D, T^3 utilizes different combinations of granularity, n-grams and score assignments to convert strings to time series as follows.

Granularity. Each record or document in text domain can be viewed as a sequence of characters or word tokens. To transform text data into time series, T^3 can use different units of text data: (1) *character level*: An alphabet letter is regarded as a single text unit. In the transformation, ignoring upper/lower cases, we consider 64 (= 26 + 10 + 28) cases – i.e., 26 cases for 26 English alphabets, 10 cases for 10 numbers (e.g., 0 to 9), and 28 cases for all special characters such as `@`, `#`, `$`. We do distinguish among special characters since some of special characters in record strings appear in our data sets; and (2) *word level*: At this granularity, an English word (also called "token") is regarded as a single text unit and T^3 simply extracts each token from sequences and then assign appropriate values to each token based on the weighting scheme.

N-grams. In statistical natural language processing, an n-gram is a subsequence of n consecutive items from a given sequence. These items could be symbols, letters, or words according to the application. As mentioned above, in our T^3 framework, we treat either a character or a token as the single unit of sequences. Therefore, an n-gram in T^3 is a sub-sequence of n consecutive "characters" at character level and a sub-sequence of n consecutive "tokens" at word level. In particular, T^3 adopts three sizes of n-grams – *unigram*, *bigram* and *trigram*. Table 1 shows an example of how to transform the record "`time series data mining`" based on different combinations of granularity level and n-grams.

Score Assignment. At this stage, T^3 assigns appropriate numeric values in order to actually convert strings to time series. Based on different levels of granularity, T^3 adopts different weighting schemes in order to assign scores to subsequences of text units. At character level, first, T^3 uses the QWERTY keyboard layout to allocate each text unit (i.e., alphabets, numbers or special alphabets) into equal-length or varying-length bins within the range of [0,1]. Then the median value of each bin is used to represent the corresponding character. During the allocation of bins, we consider the following three possible layouts: (1) *Linear order* is simply based on each key position on the keyboard following the order of row by row. Each character is then assigned an appropriate real value within the range [0,1]; (2) In the *Hilbert order*, we regard the keyboard as a small 2D space and then adopt the space-filling curve techniques to map 2D space to 1D sequence. After we get the 1D sequence, we then allocate it to uniform bins within the range [0,1] so that each character is assigned a real value. Hilbert order has good locality-preserving behaviors so that alphabets from similar locations in the keyboard layout have the similar real values during the score assignment. Our idea is motivated by the fact that alphabets from the similar locations in the keyboard have a higher probability of typo, a common issue in the record linkage problem; and (3) In addition to Hilbert space-filling curve, we also implement the *Z order* space-filling curve. Z order also has a good locality-preserving behavior similar to the Hilbert order. We are interested in whether there is a significant performance difference between these two space-filling curves.



(a) Original string records

(b)Time series after transformation

Fig. 2. A simple example of transforming text to time series.

At word level, second, T^3 uses the *tf-idf* (term frequency-inverse document frequency) weight of the traditional IR weighting scheme. The tf-idf weight is a statistical measure to estimate how important a token in a string record is within a record database such that it increases proportionally to the number of times that the token occurs in the string but is offset by its frequency in the database. Each token of a record string is assigned an importance weight using the tf-idf weight such that the whole string can be converted into a time series.

Coding	Transformation
`char + unigram`	{t,i,m,e,s,e,r,i,e,s,d,a,t,a,m,i,n,i,n,g}
`char + bigram`	{ti,im,me,es,se,er,ri,ie,es, sd,da,at,ta,am,mi,in,ni,in,ng}
`char + trigram`	{tim,ime,mes,ese,ser,eri,rie,ies,esd,sda, dat,ata,tam,ami,min,ini,nin,ing}
`word + unigram`	{time, series, data, mining}
`word + bigram`	{time series, series data, data mining}
`word + trigram`	{time series data, series data mining}

Table 1. Examples of T^3 transformation.

Discussion. Note that the three dimensions of approaches (i.e., granularity, n-gram, and score assignment) in T^3 are not exhaustive at all. One can easily devise more sophisticated transformation schemes from text strings to numeric time series. For instance, as to the score assignment dimension, in addition to the keyboard layout based assignment for the character level or weighting based assignment for the word level, one may use Linguistic characteristic (e.g., while a character-level bigram "on" occurs frequently, another bigram "xz" rarely occurs in English) to assign different assignment scores. Similarly, domain-specific characteristics of text data can be adopted. For instance, instead of character-level or word-level, one may use phrase-level or paragraph-level summary as the basic text unit when dealing with documents. Since the immediate goal of this paper is first to evaluate the validity of T^3 framework to show that "*there exist some reasonable information-lossy conversion schemes from text domain to time series domain so that text-based data mining problems can be solved in time series domain*", we rather leave the development of more sophisticated conversion schemes in T^3 framework to the future work.

4 Experimental Validation

In order to validate our proposed T^3 framework, we use the *record linkage* problem. In a nutshell, once we transform all textual records into time series data using T^3 framework, for a given query time series q, we attempt to retrieve q's true duplicate time series. Then, we compare the performance of T^3 with that of a traditional record linkage solution that uses the text string as input. If the performance of T^3 in solving the record linkage problem in time series domain is comparable to that of a traditional record linkage solution, then it shows the validity of our proposed T^3 framework. Since the transformation schemes in T^3 "lose" some information of original text string (i.e., lossy conversion), we expect the accuracy of T^3 framework to drop slightly, compared to the accuracy of a traditional record linkage solution. However, what we are more interested in these experimentations is the comparison among different schemes in T^3 framework and any possible benefits of those schemes.

Set-Up. Table 2 shows the summary of data sets that we used in our experiments. The first five data sets `map`, `bird`, `business`, `census`, and `university` are real data sets[1] which contain *real* string data and *real* errors. The data

[1] Downloaded from: http://secondstring.sourceforge.net/

Name	Data	Error	Domain	# of records	Max # of duplicates	# of queries	# of targets
`map`	real	real	map name	337	2	19	19
`bird`	real	real	bird name	982	2	67	67
`business`	real	real	business name	2,139	2	279	279
`census`	real	real	census info.	841	2	326	326
`university`	real	real	university name	116	16	15	15
`cora`	real	real	citation	1,326	5	98	194
`restaurant`	real	real	restaurant info.	864	2	111	111
`celebrity`	real	real	celebrity address	2615	2	276	276
`dblp`	real	synthetic	citation	5359	5	1,369	3,991
`dbgen`	synthetic	synthetic	mailing list	9,947	19	960	8,987

Table 2. Summary of data sets.

Name	Sample Data
`bird`	"Gavia stellata Red-throated Loon"
`business`	"3Com Corporation"
`restaurant`	"cassells 3266 w sixth st la 213 480 8668 hamburgers"
`celebrity`	"ANDRE AGASSI 8921 ANDRE DR. LAS VEGAS NV 89113"
`dblp`	"Bell Data Modelling of Scientific Simulation Proams SIGMOD Conference 1982"
`dbgen`	"Colbri P Beer 478 Naftel St 6j2 Rio Blanco PR 00744"

Table 3. Examples of some data sets.

sets `cora`, `restaurant`[2], and `celebrity`[3] are also real data sets containing real string data and real errors. We pre-processed each data set to delete some of the duplicates which were incorrectly labeled. The citation data set `dblp` was generated using real citation records from similar venues of DBLP. We randomly selected ten venues from similar research domains such as SIGMOD, PODS, and EDBT, and again randomly selected citations published in those venues. Then, we generated duplicates by injecting typographical errors. Using the data generation tool, DBGen [10], we generated one synthetic data set `dbgen` containing mailing list information. Note that unlike aforementioned data sets, this data set contains only synthetically generated data and errors. In order to get a general idea, Table 3 shows examples of record strings in some of the data sets.

As for distance measures, we use the *Levenshtein Distance* (LD) in text domain and *Euclidean Distance* (ED) and *Dynamic Time Warping* (DTW) in time series domain. All measures are known to work well for order-conscious text or time series data [1]. Since ED requires two time series to have the same length, in our experimentation, we augment the shorter time series to have the same length as the longer time series by simply adding prefix or suffix of median values (i.e., 0.5). To evaluate the efficiency and effectiveness of the proposed T^3 framework, we mainly use two evaluation metrics – speed and accuracy. In particular, to measure the speed of a method, we use the *Running Time* (T) excluding any pre-processing steps. To measure the accuracy, we use the average *Precision* (P) and *Recall* (R). Suppose that T denotes a set of true matching records and S denotes a set of records retrieved by an algorithm. Then, we have:

[2] Downloaded from: http://www.cs.utexas.edu/users/ml/riddle/

[3] Provided by Ned Porter at US Census Bureau.

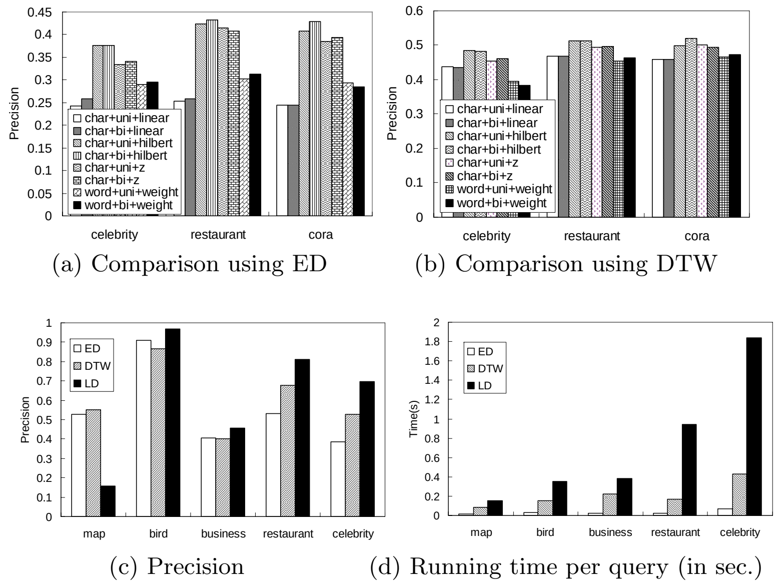
precision=$\frac{|S \cap T|}{|S|}$ and recall=$\frac{|S \cap T|}{|T|}$. We will use the precision-recall (PR) graph to present the accuracy.

Comparison of Transformation Schemes in $\mathbf{T^3}$. We first compare the performance of different combinations of granularity, n-grams and score assignment in T^3. In this comparison, we choose one distance function (i.e, either ED or DTW) and then perform tests of all major coding schemes using the same distance measure. Figures 3(a) and (b) present the precision of the record linkage task using ED and DTW, respectively. Among data sets, the results of tests on `celebrity`, `restaurant`, and `cora` are presented. In Figure 3(a) with ED, note that both Hilbert and Z order based schemes outperform the others with respect to the precision. This is reasonable because both Hilbert and Z order have a good locality-preserving behavior such that alphabets in the neighborhood in the keyboard have the similar real values during the score assignment. Therefore, this can reduce a number of false positives in cases when true duplicates have some dissimilar characters caused by typos or data entry errors. Between these two orders, Hilbert order performs slightly better than Z order scheme, but not significantly. Figure 3(b) shows the similar results when DTW is adopted as the distance function in the experiment.

Another interesting finding is that the word-level transformation schemes using tf-idf weighting as scores do *not* show a significantly better precision, although they can find true duplicates faster because of the shorter time series generated. The reason is that using the word-level schemes based on tf-idf weights, the resulting time series is entirely determined by the tf-idf weight of each token. To some extent, we lose the lexical information of tokens. For instance, there might exist a situation where two tokens are completely different but happen to carry equal or similar tf-idf weights. This can affect the shapes of time series and hence generate false positives.

Also note that from Figures 3(a) and (b), we do not see much difference between unigram and bigram schemes (trigram schemes have similar patterns and not shown for limited space). This is partially because our record linkage solutions are obtained in real-value domain after record stings are converted to time series, and higher-gram techniques may not be as effective as in the case of string manipulation in text domain. Overall, the transformation scheme based on the combination of character-level granularity, unigram and Hilbert order appears to be the best scheme. Therefore, we adopt this scheme (denoted as `char-uni-hilbert`) in the following experiments.

Comparison of Distance Functions with Baseline. Next, we compare the performance of different distance functions in our proposed T^3 framework. In this comparison, we fix the transformation scheme to `char-uni-hilbert` and then compare among ED and DTW (in time series domain), and LD (in text domain) as the baseline. Figures 3(c) and (d) show the precision and running time of three distance measures in the context of record linkage problem. The results of tests on `map`, `bird`, `business`, `restaurant` and `celebrity` are presented. In these data sets, each query string has exactly one duplicate. Therefore record linkage



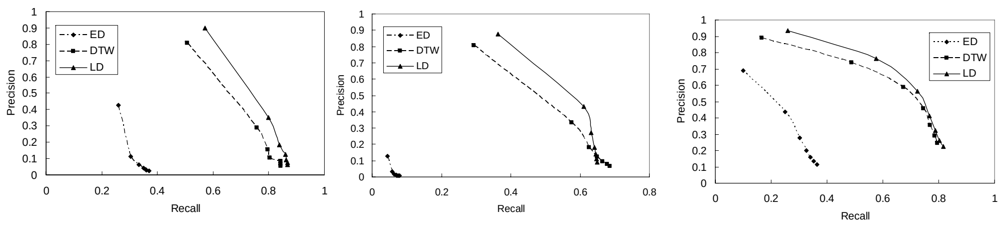
(a) Comparison using ED (b) Comparison using DTW

(c) Precision (d) Running time per query (in sec.)

Fig. 3. (a)-(b) Comparison among eight different transformation schemes based on distance function ED and DTW using three data sets. (c)-(d) Comparison among ED, DTW, and LD using five data sets (based on `char-uni-hilbert`).

on these data sets is straightforward and aims to find the other duplicate for each of the query records.

In Figure 3(c), note that LD consistently produces better precision than ED and DTW, except `map` data set. This is as expected because LD directly operates on the original record strings in text domain without the loss of any information. What we are more interested is: *as a complementary approach for solving the record linkage problem in time series domain, how good is the performance of T^3 techniques compared to the baseline?* Figure 3(c) shows that T^3 with ED and DTW can yield comparable precision on four data sets (and better precision on one data set). Since the various transformation schemes in T^3 tend to lose some information from original text strings during the conversion, we expect to lose some degree of accuracy in T^3, when compared to LD. Therefore, although there appears to be degraded accuracy in T^3, since it is comparable to the baseline without using T^3, we believe that the result is still promising. Also note that the overall precision of either our proposed schemes or the baseline is around 0.6 across all the data sets in Figure 3(c). This is due to the characteristics of our data sets. As shown in Table 3, our data sets are real data sets which contain a lot of mis-spelling errors and mis-alignments. Therefore, we expect low degree of accuracy of matching similar records. More research is needed so that one can transform text to time series while maintaining or improving the accuracy. Another interesting finding is that DTW mostly performs better than ED (i.e., in `map`, `restaurant`, and `celebrity` data sets) and the difference increases as



(a) cora (b) dblp (c) dbgen

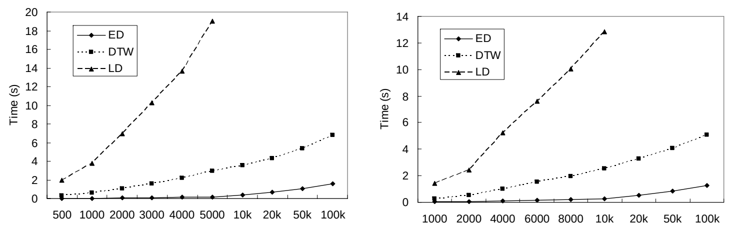
Fig. 4. PR graphs of ED, DTW and LD methods using `char-uni-hilbert`.

the size of data sets and lengths of record strings increase. This is reasonable as DTW is usually regarded as a much more robust distance measure for time series [1] and allows similar shapes to match even in the case that two time series are not aligned well in the time axis.

Figure 3(d) shows the average running time per query. We can clearly see that both ED and DTW methods consistently run much faster than the baseline LD, across all data sets. Furthermore, ED method even performs faster than DTW. While ED is the fastest (with low accuracy), overall, DTW shows a good trade-off of having faster running time with comparable accuracy. The running time of DTW increases as the length of record strings increases. This is partially due to the cost of dynamic programming in DTW, which is in the magnitude of square of record length.

The precision-recall (PR) graphs in Figure 4 are generated with both precision and recall of the ED, DTW and LD methods by increasing k, which is the number of answers returned by an algorithm to solve the record linkage task. The value of k changes from 1 to 30. At each point, corresponding precision and recall values are measured and plotted. The PR graphs of three large data sets, `cora`, `dblp` and `dbgen`, are presented. As we can see from Figure 4, DTW and the baseline LD outperform ED by a large margin. Furthermore, DTW produces PR curves that are comparable to the baseline. This is consistent with what we found in Figure 3(c). In addition, note that both LD and DTW run much faster than LD (in Figure 5), again consistent with Figure 3(d).

Scalability. Since the scalable processing is a critical issue in the record linkage problem, we also study the scalability of T^3 framework. We select two large data sets, i.e., `dblp` and `dbgen`. From `dblp` with 5,359 real citation records, we prepare different subsets of 500, 1000, 2000, 3000, 4000 and 5000 records. Furthermore, we generate four different datasets of 10,000, 20,000, 50,000 and 100,000 records from the original `dblp` dataset. Also, from `dbgen` with 9,947 mailing lists, we generate subsets of 1000, 2000, 4000, 6000, 8000, and 10000 records (we add 53 more mailing lists to generate 10000 records). Again, we generate three different datasets of 20,000, 50,000 and 100,000 records from the original `dbgen` dataset.



(a) Scalability on `dblp` (b) Scalability on `dbgen`

Fig. 5. Running time per query (s) on different sizes of `dblp` and `dbgen` subsets.

With these data sets and the fixed transformation scheme of `char-uni-hilbert`, we measure the running time as the data size increases.

Figure 5 shows the results. In general, Figures 5(a) and (b) show similar patterns. As the size of data sets increases, the running time per query increases linearly for both ED and DTW. However, the running time for LD increases more rapidly compared to that of our approaches. This indicates that our record linkage solution is more scalable to handle a large amount of data. Furthermore, ED consistently outperforms DTW in terms of speed. This is as expected because DTW involves a procedure of dynamic programming in calculating the distance, which decreases the overall speed as the size of data increases. But as we mentioned earlier, DTW method has better precision in terms of accuracy of record linkage.

5 Conclusion and Future Work

In this paper, we present our preliminary design of the T^3 framework to transform text to time series data. We propose two variations of granularity, three variations of n-grams, and four variations of score assignments based on space-filling curve techniques for characters or tf-idf weighting technique for tokens. We adopt two similarity measures, Euclidean Distance (ED) and Dynamic Time Warping (DTW), to calculate the distance between two time series and show the efficacy of our proposed schemes using both real and synthetic data sets.

In terms of record linkage, our schemes in the T^3 framework show promising results with good combination of speed and accuracy, compared to conventional string matching methods such as Levenshtein Distance (LD). In particular, Hilbert space-filling technique at character-level granularity is the best variation of transformation schemes while DTW is a better distance measure regarding precision and ED outperforms regarding running time. With respect to accuracy and speed, the experimental results confirm that our T^3 framework can generate precision-recall curves comparable to the baseline LD. We believe our approach can shed new insights in both areas of text mining and time series mining.

Many future research directions are ahead. First, we plan to extend T^3 framework to other text mining areas such as document clustering and classification.

The sizes and dimensions of the data increase dramatically when documents are considered. Second, more sophisticated transformation schemes and advanced similarity functions need to be devised to provide comparable accuracy using time series data to their counterpart using text data.

The implementations and test data sets used in this paper are publicly available at: `http://pike.psu.edu/download/amw09/`

References

1. D. J. Berndt and J. Clifford. “Using dynamic time warping to find patterns in time series”. In *KDD Workshop*, 1994.
2. M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. “Adaptive Name-Matching in Information Integration”. *IEEE Intelligent System*, 64, 2003.
3. K. P. Chan and A. W.C.Fu. “Efficient Time Series Matching by Wavelets”. In *ICDE*, 1999.
4. S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. “Robust and Efficient Fuzzy Match for Online Data Cleaning”. In *SIGMOD*, 2003.
5. H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh. “Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures”. In *VLDB*, 2008.
6. A. Elmagarmid, P. Ipeirotis, and V. Verykios. “Duplicate Record Detection: A Survey”. *TKDE*, 19(1), 2007.
7. C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. “Fast Subsequence Matching in Time-Series Databases”. In *SIGMOD*, 1994.
8. I. P. Fellegi and A. B. Sunter. “A Theory for Record Linkage”. *J. of the American Statistical Society*, 18(5), 1969.
9. L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. “Text Joins in an RDBMS for Web Data Integration”. In *WWW*, 2003.
10. M. A. Hernandez and S. J.Stolfo. “The Merge/Purge Problem for Large Databases”. In *SIGMOD*, 1995.
11. Y. Hong, T. Yang, J. Kang, and D. Lee. “Record Linkage as DNA Sequence Alignment Problem”. In *QDB*, 2008.
12. E. J. Keogh, K. Chakrabarti, M. J.Pazzani, and S. Mehrotra. “Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases”. *Knowl. Inf. Syst.*, 3(3), 2001.
13. J. Lin, E. J. Keogh, L. Wei, and S. Lonardi. “Experiencing SAX: a novel symbolic representation of time series”. *Data Min. Knowl. Discov.*, 15(2), 2007.
14. B. W. On, D. Lee, J. Kang, and P. Mitra. “Comparative Study of Name Disambiguation Problem using a Scalable Blocking-based Framework”. In *JCDL*, 2005.
15. C. A. Ratanamahatana and E. J. Keogh. “Three Myths about Dynamic Time Warping”. In *SDM*, 2005.
16. S. Sarawagi and A. Bhamidipaty. “Interactive Deduplication using Active Learning”. In *KDD*, 2002.
17. M. Vlachos, D. Gunopulos, and G. Kollios. “Discovering similar multidimensional trajectories”. In *ICDE*, 2002.
18. J. W. Warnner and E. W. Brown. “Automated Name Authority Control”. In *JCDL*, 2001.
19. W. E. Winkler. “The State of Record Linkage and Current Research Problems”. Technical report, US Bureau of the Census, 1999.