Dynamic Time Warping

Andriciuc Andreea, Dinu Florin, Tudose Ștefan

31 ianuarie 2024

1 Introducere

Într-o lume dominată de tot felul de dispozitive smart (*Internet of Things*), avem nevoie de metode de a interpreta semnalele acestor dispozitive amplasate în medii diverse: acasă, mall-uri, aeroporturi etc. Un exemplu pertinent în acest sens îl reprezintă orice asisitent vocal inteligent care trebuie să detecteze și să interpreteze comenzi vocale, chiar dacă utilizatorul vorbește mai încet sau mai alert. Unul dintre cei mai comuni algoritmi prin care acest lucru este posibil se numește *deformarea dianmică a timpului* (în engleză Dynamic Time warping, prescurtat DTW). Algoritmul efectuează o aliniere sistematică a seriilor de timp prin care își propune să genereze o metrică reprezentativă și precisă a apropierii dintre ele.

În consecință, el pavează drumul pentru nenumarate aplicații în domeniul Procesării Semnalelor, câteva dintre acestea fiind identificarea tiparelor în serii de timp [1], detectarea anomaliilor[2], și învățarea automată, în special cea nesupervizată prin generarea de clustere datorate scorului de similaritate furnizat de către algoritm (prin utilizarea abordării k-Nearest Neighbours)[3].

Alături de prezentarea unei perspective generale asupra problematicii comparărilor seriilor de timp și mecanismul prin care *deformarea dinamică a timpului* constituie o soluție la aceste dileme, lucrarea de față își propune să exploreze o serie de rezultate comparative empirice asupra câtorva dintre domeniile de aplicabilitate ale algoritmului, și anume:

- date din situațiile financiare,
- comparație între salariile medii,
- text mining,
- detectarea similarităilor în melodii celebre.

2 Principiu de funcționare

Orice două serii de timp pot fi comparate, cel mai comun, folosind distanța euclidiană (Figura 1). Amplitudinea primei serii de timp la momentul t va fi comparată cu amplitudinea celei de-a doua serii tot la momentul t, ceea ce poate rezulta în rezultate neconcordante. În plus, nu toate semnalele pot fi aliniate unu la unu (Figura 2).

Deși folosită inițial pentru potrivirea de texte și găsirea similarităților, metoda

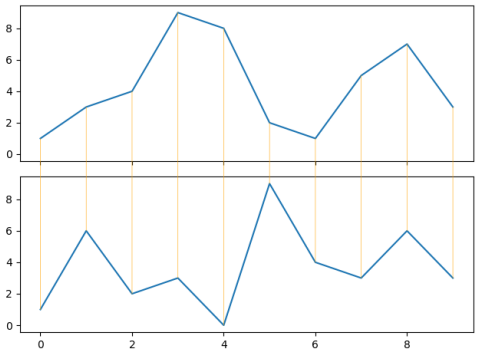

Figura 1: Distanța euclidiană.

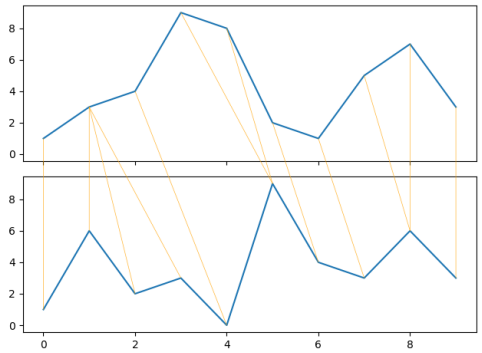

Figura 2: Distanța DTW.

deformării dinamice a timpului este în prezent cunoscută și ca mod de măsurare al distanței dintre serii de timp. Un exemplu simplu de potrivire după aplicarea DTW pentru două serii de timp se găsește în Figura 2.

2.1 Abordare teoretică (matematică)

Considerând două serii de timp P = $\{p_1, p_2, \ldots, p_n\}$ și Q = $\{q_1, q_2, \ldots, q_n\}$ de lungime n și, respectiv, m, construim matricea de distanțe, $D = (d(p_i, q_j))_{n*m}$, unde $d(p_i, q_j)$ reprezintă distanța inițială dintre punctele p_i și q_j. Metoda DTW caută să „alinieze" cele două serii într-un mod care minimizează distanța euclidiană între fiecare pereche de puncte. Vrem, deci, un drum optim R = $\{r_1, r_2, \ldots, r_K\}$, care să minimizeze acumularea de distanțe între puncte, unde $max(n, m) \leq K < m + n - 1$. Impunem următoarele constrângeri pentru drumul optim:

1. *Condiția de graniță:* Primul și ultimul punct din prima serie de timp vor fi corelați cu primul și ultimul punct din a doua serie, adică $r_1 = d(p_1, q_1)$ și, respectiv, $r_K = d(p_n, q_m)$;

2. *Condiția de monotonie:* Pentru $r_k = d(i_k, j_k)$ și $r_{k+1} = d(i_{k+1}, j_{k+1})$, avem $i_{k+1} \geq i_k$ și $j_{k+1} \geq j_k$;

3. *Condiția de pas:* $i_{k+1} \leq i_k + 1$ și $j_{k+1} \leq j_k + 1$

Mai formal, problema ia următoarea formă:

$$DTW(P,Q) = \min_{w}(\sum_{k=1}^{K} dist(p_i, q_j))$$

[7]

2.2 Implementare generică

Un aspect interesant legat de acest algoritm este faptul că, deși problema este una de minimizare peste o mulțime finită, numărul de drumuri admisibile crește extrem de mult chiar și pentru lungimi moderate ale celor două serii. Presupunând n≈m, există $O(\frac{(3+2\sqrt{2})^n}{\sqrt{n}})$ drumuri admisibile (număr Delannoy). O soluție care nu pierde din precizie a fost găsită folosind metoda programării dinamice. [6]

```
int DTWDistance(s: array [1..n], t: array [1..m]) {
    DTW := array [0..n, 0..m]
    for i := 0 to n
        for j := 0 to m
            DTW[i, j] := infinity
    DTW[0, 0] := 0
    for i := 1 to n
        for j := 1 to m
            cost := d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j ], // insertion
                                        DTW[i  , j-1],   // deletion
                                        DTW[i-1, j-1])   // match
    return DTW[n, m]
}
```

Sursă: Wikipedia

2.3 Exemplu concret

Vom urmări exemplul pentru care am și generat Figura 2, la care se adaugă generarea matricii de distanțe, în figura 3. Astfel, avem cele două serii de timp: P = [1,3,4,9,8,2,1,5,7,3] și Q = [1,6,2,3,0,9,4,3,6,3] Calculăm, începând cu punctul (0, 0) (în reprezentarea din Figura 3 colțul din stânga sus) fiecare element al matricei, folosind formula:

$$D(i,j) = dist(p_i, q_j) + \min\{D(i-1, j-1), D(i, j-1), D(i-1, j)\},$$

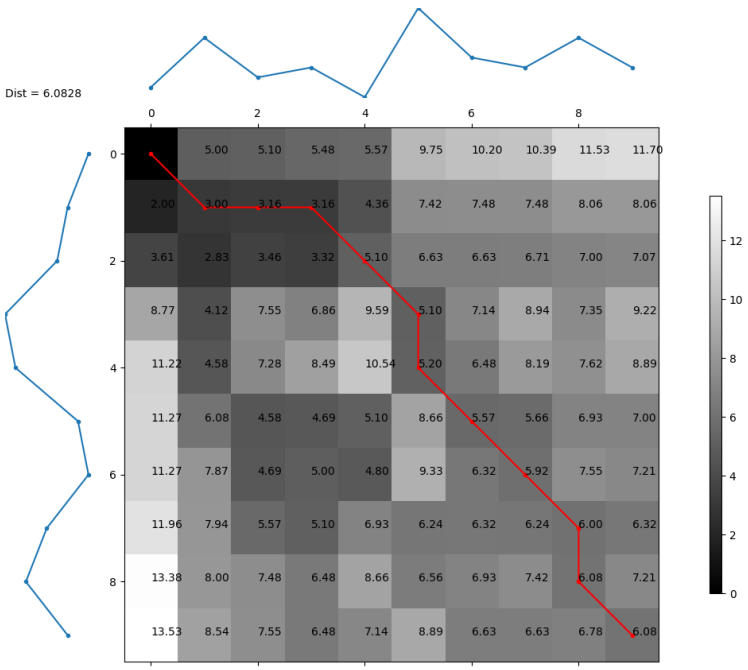

Figura 3: Exemplu matrice de distanțe.

unde $dist(p_i, q_j)$ reprezintă o funcție de distanță (în majoritatea cazurilor, inclusiv acum, distanța euclidiană).

După completarea matricei, trebuie să identificăm drumul optim, începând de la ultimul element completat (în acest caz, elementul din dreapta jos). Drumul optim este identificat bazat pe valoarea minimă a vecinilor elementului curent. Mai exact, dacă la pasul k avem elementul r_k, aflat pe pozitia (i, j), pentru următorul pas vom alege dintre vecinii lui r_k, selectând $\min\{D(i, j-1), D(i-1, j), D(i-1, j-1)\}$. Drumul final pentru exemplul nostru este evidențiat printr-o linie roșie.

3 Experimente

3.1 Implementări/Optimizări

Pentru început, am încercat să implementăm manual un algoritm DTW, ca mai apoi să găsim foarte multe bibioteci care implementează deja, optimizat, aceeași metodă. Ca prim experiment, ne-am propus să analizăm eficiența funcțiilor pre-implementate comparat cu funcția pe care am definit-o noi. Pentru a realiza acest lucru, am definit două semnale sinusoidale defazate cu 3200 de eșantioane, pentru care am cronometrat, folsind modulul *time*, în cât timp obținem distanța și, respectiv, drumul optim raportate la seriile de timp definite. Rezultatele se pot observa în tabelul 1.

Mai apoi, am încercat să analizăm importanța parametrilor pe care anumite

Implementare	Timp de rulare (secunde)
Manual (*)	20.144052267074585
dtaidistance (*)	0.0877840518951416
tslearn (*)	1.1982133388519287
pyts (*)	1.016982078552246
DTW-python (*)	0.29955387115478516

Tabela 1: Comparație de eficiență (Implementare).

funcții din aceste bibioteci le implementează și ce efect au acestea asupra timpului de rulare și, respectiv, al rezultatului final. Astfel, pentru același set de sinusoide cu 3200 de eșantioane, am adăugat codului o condiție suplimentară ce restricționează calculul matricei de distanțe doar la un sector de diagonale la diverse distanțe față de diagonala principală. Rezultatele sunt vizibile în Figurile **??**.

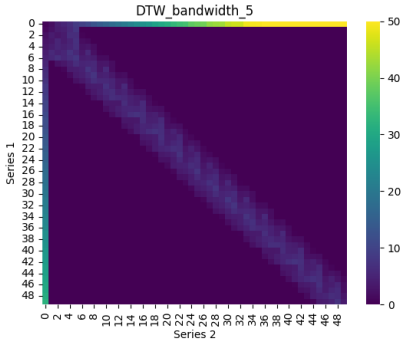

Figura 4: Bandwidth = 5.

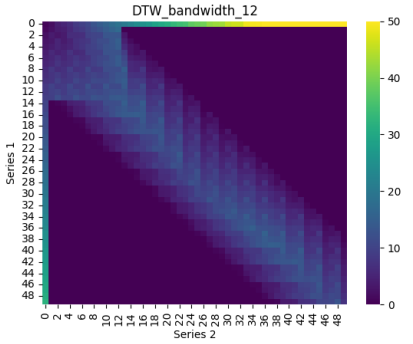

Figura 5: Bandwidth = 12.

Am obținut următoarele rezultate:

- Rulând algoritmul simplu, am obținut o distanță de 45.23.
- Rulând algoritmul cu o fereastră de 10%, am obținut o distanță de 3.53
- Rulând algoritmul cu o fereastră de 25%, am obținut o distanță de 7.27

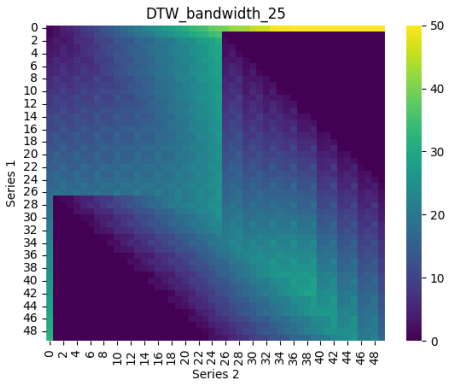

Figura 6: Bandwidth = 25.

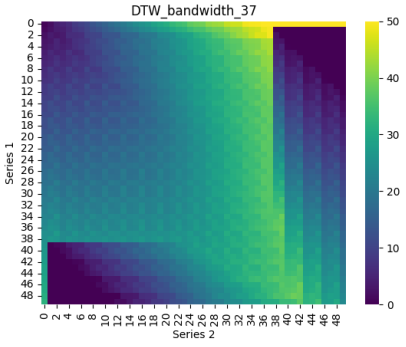

Figura 7: Bandwidth = 37.

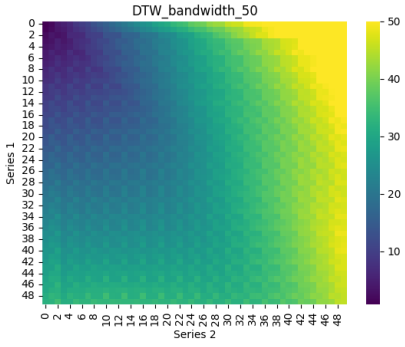

Figura 8: Bandwidth = 50.

- Rulând algoritmul cu o fereastră de 50%, am obținut o distanță de 16.29
- Rulând algoritmul cu o fereastră de 75%, am obținut o distanță de 23.57
- Rulând algoritmul cu o fereastră de 100%, am obținut o distanță de 45.23, la fel ca pentru algoritmul naiv.

3.2 Comparație între salariile medii

Un alt experiment intersant a fost găsirea unor similarități privind evoluția salriilor medii din țări membere Organizației pentru Cooperare și Dezvoltare Economică (OCDE). Folosind datele puse la dispoziție de aceștia cuprinzând date salariale din perioada 2000-2022, am reușit să aflăm că Estonia și Lituania sunt țările care au avut cea mai similară evoluție a salariilor medii, urmate îndeaproape de Finlanda cu Germania. Rezultatele se pot observa și în Figura 9.

3.3 Determinare DTW între sumele înregistrate ca Venit, Costul Vânzărilor, Profit Brut, Cheltuieli de Exploatare și Profit Înainte de Dobânzi și Taxe

Tot din zona fiscală, am găsit un set de date pentru valorile înregistrate de o firmă în decursul anilor 2019-2020. Întrebarea a fost dacă există vreo corelație între sumele înregistrate, iar răspunsul a fost, aproape indiscutabil, nu, după cum se poate observa și din Figura 10.

3.4 Comparații pe texte

De asemenea, ne-am propus și un experiment puțin mai ... îndrăznet, și anume să facem o comparație între folosirea cuvântului "Dumnezeu"/"God"/"Gott" [5] în ferestre constând în capitole din Bibliile traduse în română (traducere Cornilescu), engleză (King James) și germană (Luther). Inițial, căutăm numărul de apariții din fiecare capitol pentru fiecare dintrte cele trei exemple. Reprezentând date astfel

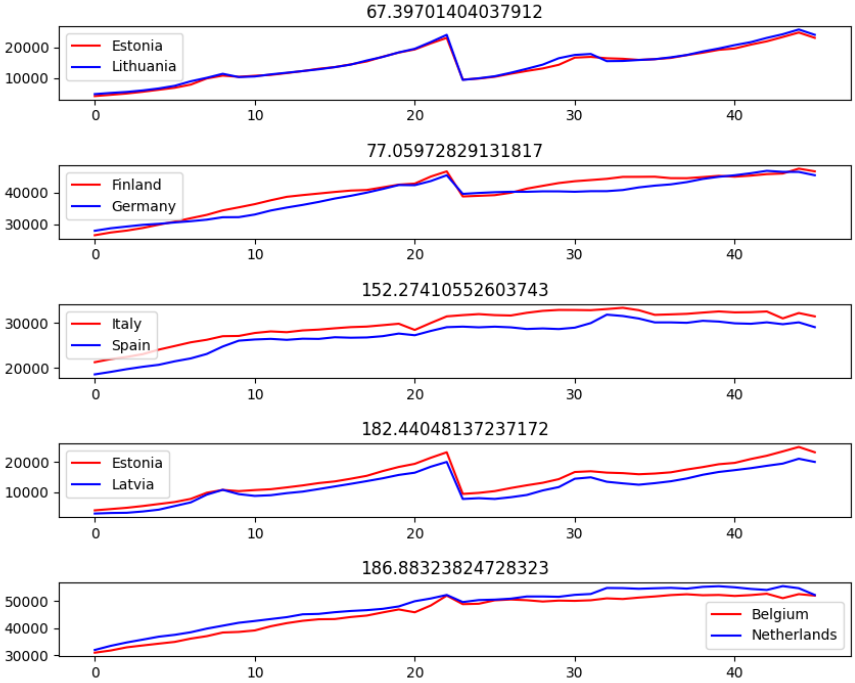

Figura 9: Cele mai similare 5 țări.

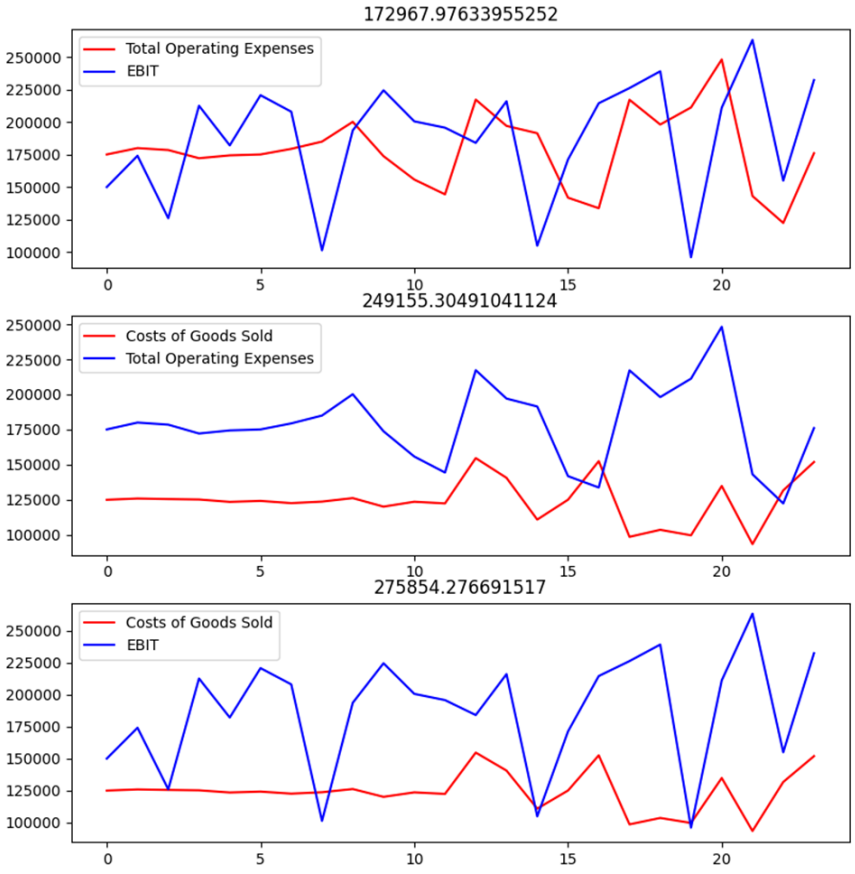

Figura 10: Date financiare.

extrase, obținem graficele din figurile 11, 12 și, respectiv, 13. Intuiția din spate este că pentru fiecare apariție a cuvântului „Dumnezeu” în română ar trebui să existe o apariție a cuvintelor „God” sau „Gott” în engleză și, respectiv, germană, în poziții aproximativ similare. Cu toate acestea, pot apărea discrepanțe între aceste apariții din diverse motive (sintaxa diferită, interpretare subiectivă etc). În final, am obținut următoarele distanțe:

- distanța DTW engleză-română: 1.105296198989212,
- distanța DTW engleză-germană: 1.24358771973137,
- distanța DTW română-germană: 1.3280902773238819,

adică cea mai mare similaritate este între bibliile din română și engleză, deși ne așteptam să găsim o similaritate mai mare între versiuna în engleză și cea în germană, intuiție bazată pe cunoașterea faptului că ambele sunt limbi germanice.

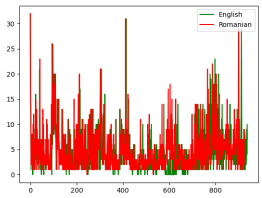

Figura 11: Română vs Engleză.

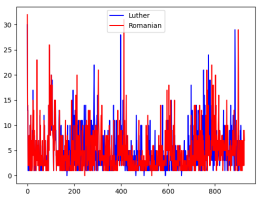

Figura 12: Română vs Germană.

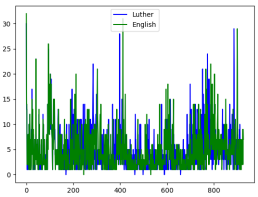

Figura 13: Germană vs Engleză.

3.5 Analiză muzicală

O altă utilizare interesantă a metodei DTW este aceea că poate detecta similarități și între sunete, mai precis, melodii. Exemplul ales constă în patru fișiere .mp3, conținând exclusiv versiuni instrumentale pentru „Every breath you take”, „Every breath you take, slowed and reverbed” și „I'll be missing you”. În plus, pornind de la primul fișier, am mai adăugat un exemplu, modificat manual în Audacity pentru a-i mări tempo-ul și a-i mai adăuga efecte precum ecou, reverb etc.

Pentru prelucrarea fișierelor audio am folosit biblioteca *librosa*[4], care oferă funcționalități precum: extragere de caracteristicilor, analiză spectrală și, cel mai important pentru analiza noatră, modelare secvențială. Toate fișierele au o frecvență de eșantionare de 22050, dar dimensiunile lor diferă, fiind, în ordine, (5534721,) (6534144,) (3320833,) (6808576,). Spectogramele lor sunt reprezentate în Figura 14.

Se observă că a doua reprezentare este mult mai „întinsă”, a treia păstrează sunetele înalte, dar le egalizează pe cele mai joase, iar ultima este considerabil diferită, mai ales ca fluctuații. Folosirea decibelului ca unitate de măsură în reprezentarea grafică a sunetului este utilă din prisma faptului că acesta este la scară logaritmică,

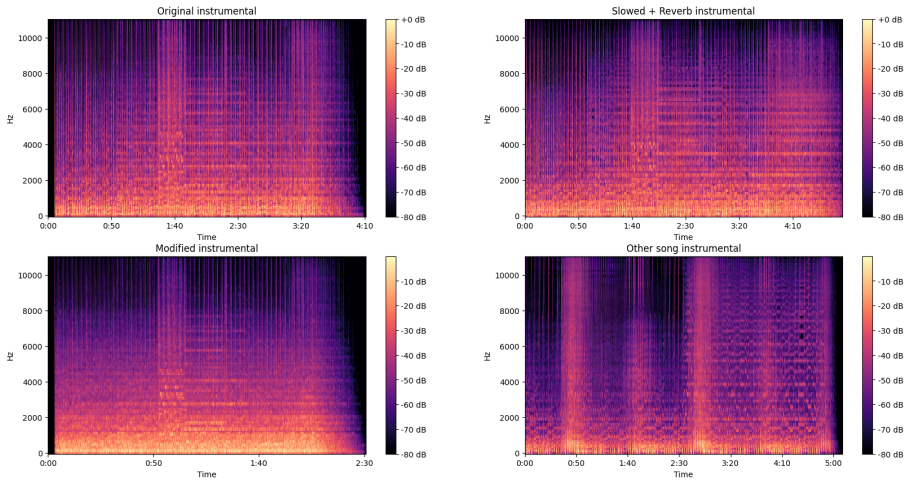

Figura 14: Spectograme.

ceea ajută la reprezentarea puterii sunetului, fiind o practică standard când vine vorba de procesarea de semnale audio.

Aplicarea metodei DTW pentru cele patru instanțe folosite produce graficele din Figurile 15, 17, 16.

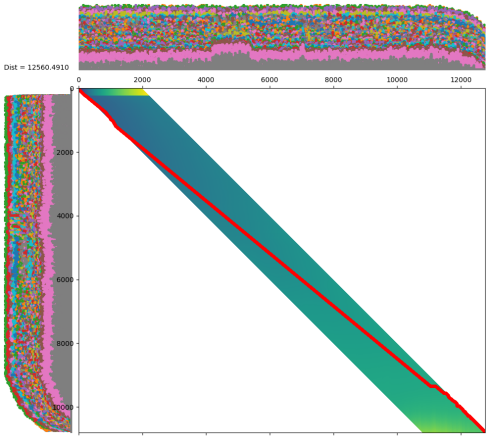

Figura 15: Original vs Slowed&Reverebed.

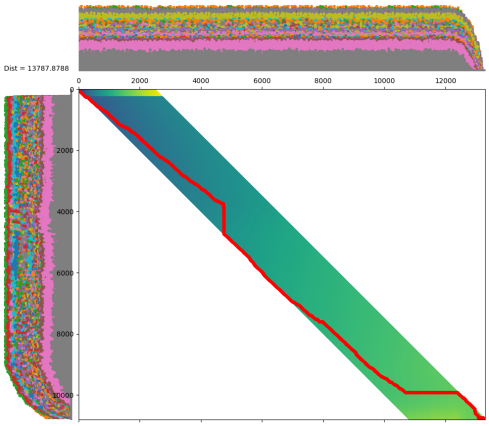

Figura 16: Original vs Other.

Observăm că există cele mai multe similatități între versiunea originală și cea modificată manual în Audacity, unde singura diferență considerabilă este, pe lângă lungimea celor două piese, este în partea de început și, respectiv, de final, unde probabil există niște timpi morți. De asemnenea, după cum ne așteptam, cea mai mare diferență este între piesa originală și piesa complet diferită, deși valoarea obținută ca distanță este oarecum apropiată ca și diferența dintre piesa originală și varianta modificată profesional.

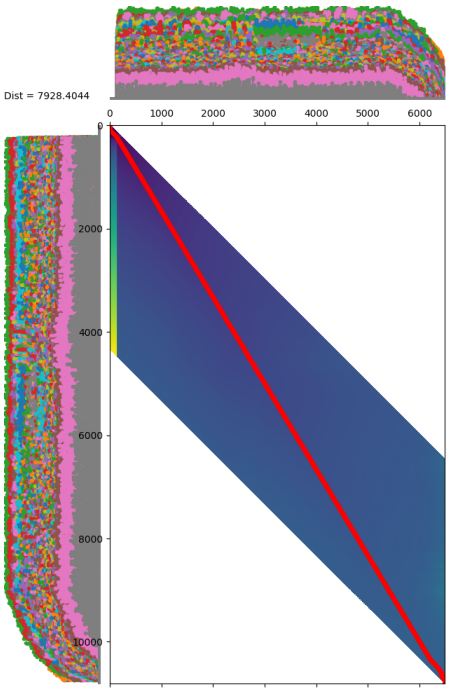

Figura 17: Original vs Modified.

4 Concluzii

În această lucrare am reușit să ne familiarizăm cu conceptul de deformare dinamică a timpului și să înțelegem în profunzime aplicațiile practice ale acestei metode. DTW este o tehnică larg folosită pentru orice se poate modela ca o serie de timp, cu prea puține restricții. Am reușit, astfel, să facem analiză pe date salariale, pe datele contabile ale unei firme, pe variante de traduceri ale bibliei și, nu în ultimul rând, pe melodii pentru a verifica dacă una dintre ele este plagiată (nu de noi, ci de artiști ;)). Considerăm că pot apărea mult mai multe aplicații ale acestui algoritm în viitor, dar pentru acest curs am ales să ne oprim doar la experimentele menționate anterior.

Bibliografie

[1] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. 1994.

[2] IEEE. Anomaly detection using dynamic time warping. 2019.

[3] Vivek Mahato, Martin O'Reilly, and Pádraig Cunningham. A comparison of k-nn methods for time series classification and regression. 2019.

[4] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[5] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270, 2012.

[6] Romain Tavenard. An introduction to dynamic time warping. https://rtavenar.github.io/blog/dtw.html, 2021.

[7] Yuan Wan, Xiao-Li Chen, and Ying Shi. Adaptive cost dynamic time warping distance in time series analysis for classification. *Journal of Computational and Applied Mathematics*, 319:514–520, 2017.