Felhasználók azonosítása gépelésük alapján

Feladat vezető:
Dr. Horváth Győző

Készítette:
Cseh Dávid

Budapest, 2020

A kutatást az **„Integrált kutatói utánpótlás-képzési program az informatika és számítástudomány diszciplináris területein” (EFOP-3.6.3-VEKOP-16-2017-00002)** című projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

Tartalom

Bevezetés

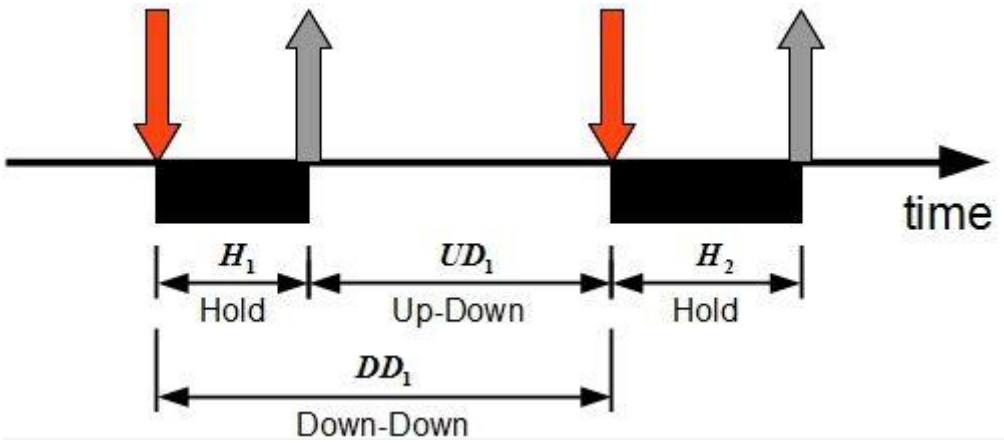
A kutatásom fő célja az volt, hogy kidolgozzak egy olyan módszert, amivel a felhasználókat azonosítani lehet a billentyűzeten való gépelésük alapján. Itt fontos megjegyezni, hogy nem statikus szövegek alapján történik az azonosítás, hanem az a célom teljesen szabad szöveg után is képes legyen az algoritmus az azonosításra, miután kellő mennyiségű adatgyűjtés történt előtte. Az ember írása a billentyűzeten is egyedi, meg lehet adni a gépelésnek olyan jellemzőit, amivel megkülönböztethetők és azonosíthatók a felhasználók. Ezzel a témakörrel sokat foglalkozott a szakirodalom, én kifejezetten a k – legközelebbi algoritmust fogom használni.

Billentyűleütések vizsgálata

A billenytűleütések vizsgálatát más néven leütés dinamikának vagy írás dinamikának is nevezik. Azzal foglalkozik, hogy pontosan leírja melyik billentyű lett leütve, időben mikor lett megnyomva és felengedve amikor egy személy éppen gépel egy számítógép billentyűzetén és olyan metódusokat ad, amivel képesek vagyunk az adott felhasználókat csak a gépelésük alapján megkülönböztetni és azonosítani. Az emberi gépelés a billentyűzeten ugyanúgy egyedi, megkülönböztethető és használható biometrikus azonosításhoz, mint a már bevált ujjlenyomat leolvasás vagy a retina szkenner. A gépelést sok minden teheti egyedivé, például mennyi ideig volt lenyomva egy adott billentyű vagy mennyi idő telt el két billentyű leütése között. Ezek az információk lehetnek egyediek különböző billentyűk esetében, lehet valaki az 'A' betűt mindig gyorsabban üti le, mint a 'D' betűt. Az is különbség lehet, ha valaki mindig csak a bal shift-et használja a jobb oldali helyett vagy caps lock-ot használ a nagybetűkhöz.

Attribútumok

A nyers adatokból különböző attribútomokon keresztül lehet kinyerni az információt.



A hold time azt az időt jelenti, hogy mennyi ideig tartunk egy adott billentyűt lenyomva, tehát a down és az up esemény között eltelt idő.

Flight time-nak nevezzük azt az időt, ami két billentyű leütés között telik el. Tehát az első billentyű felengedése és a következő billentyű leütése között eltelt idő, jelen esetben H1-nek az up eseménye és H2 down eseménye közötti idő.

A Digraph, amikor az előző kettő kombináljuk, tehát az első billentyű nyomvatartási ideje és az első és a második billentyű között eltelt idő.

A Trigraph, ugyanaz mint a Digraph, csak itt nem kettő, hanem három egymás követő billentyűt vizsgálunk.

A további változat, az Ngraph, ahol N darab egymás követő billentyűt vizsgálunk.

K-legközelebbi szomszéd algoritmus

A következő fejezetben ismertetem a használt algoritmust.

k-LSZ(k: egész szám, a: vektor, b: (vektor, címke) párok halmaza)

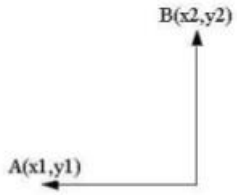
1. c := Számítsuk ki ’a’ vektorhoz az összes ’b’-beli vektor távolságát.
2. Állítsuk ’c’-t növekvő sorrendbe.
3. Vegyük ’c’-nek az első ’k’ darab elemét.
4. e := Válasszuk ki ’c’-ből a ’címke’ szerinti legtöbb előfordulást.
5. Visszatérés ’e’-vel.

Távolságmérés függvények

A k-legközelebbi algoritmusnál a vektorok távolságához az euklideszi és a manhattan távolság függvényeket fogom használni.

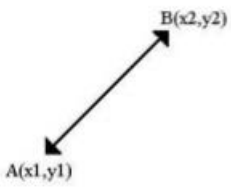
Euklideszi távolság az A(x1,y1) és B(x2,y2) pontok között:

$$D_M = |x2 - x1| + |y2 - y1|$$



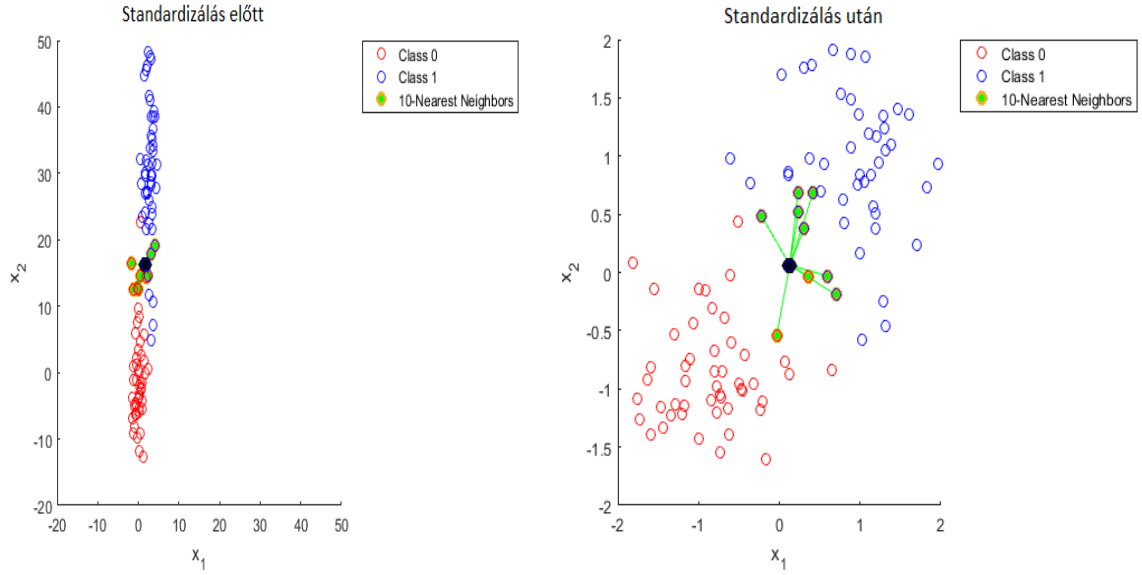
Manhattan távolság az A(x1,y1) és B(x2,y2) pontok között:

$$D_E = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

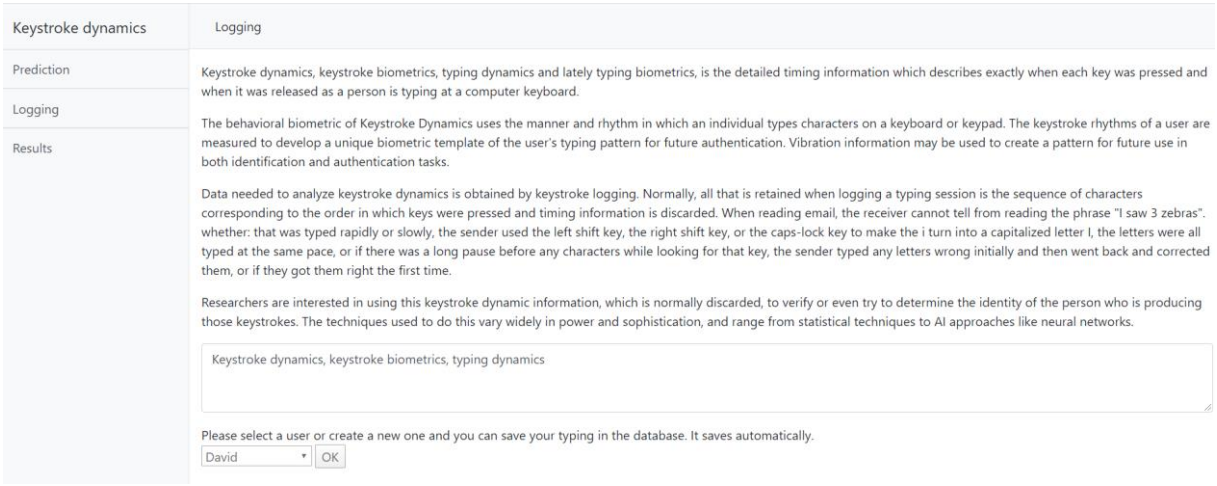


Standardizálás

Az adatok előfeldolgozásnál alkalmazva a standardizálást a k – legközelebbi szomszéd algoritmus hatékonyságát nagyban megnöveli, ezt a következő példa jól szemlélteti:



Adatgyűjtés



Az adatgyűjtést egy webes felületen valósítottam meg.

Minden billentyűleütés után létre jön két esemény, az egyik a lenyomás pillanatában, a másik a billentyű felengedésekor. Ha folyamatosan lenyomva van tartva egy billentyű akkor úgynevezett repeat események jönnek létre, ezeket nem fogom használni később. Minden eseményhez rögzítem még a pontos időt is és azt is, hogy melyik felhasználóhoz tartozik. Ezeket az adatokat tárolom nyers formában

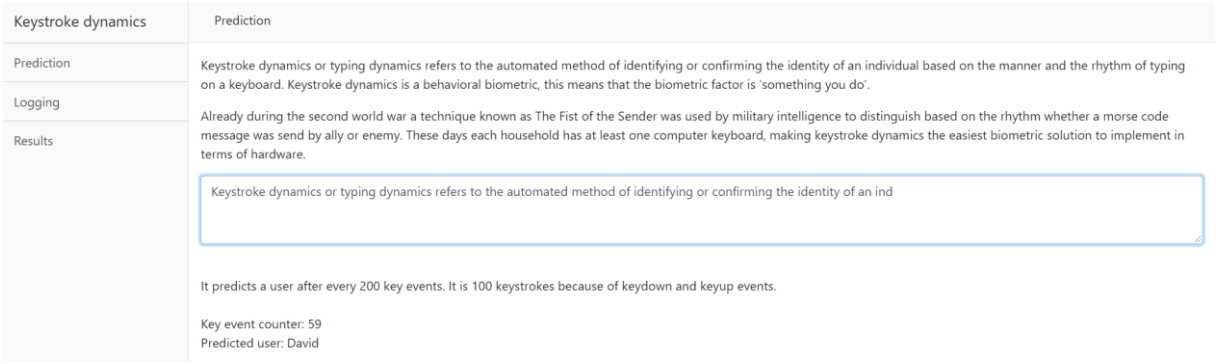
id	uid	event	keyname	keycode	timestamp
251	9	keydown	Shift	16	1574086351914
252	9	keydown	K	75	1574086352194
253	9	keyup	K	75	1574086352289
254	9	keyup	Shift	16	1574086352317
255	9	keydown	e	69	1574086352415
256	9	keydown	y	89	1574086352500
257	9	keyup	e	69	1574086352515
258	9	keyup	y	89	1574086352595
259	9	keydown	s	83	1574086352992

Nyers adatok tárolt formában

Előfeldolgozás

Az előfeldolgozás során a nyers adatokból elsőnek kiszűröm felesleges repeat adatokat, majd előállítom a sorrendben az ugyanahhoz a billentyűhöz tartozó lenyomás és felengedéseket eseményeket. Ebből előállítok olyan sorozatot, amiknek az elemei a jellemző attribútumok, ami lehet flight time, hold time, digraph, trigraph, ngraph sorozatok. Ez után következik a strandardizálás.

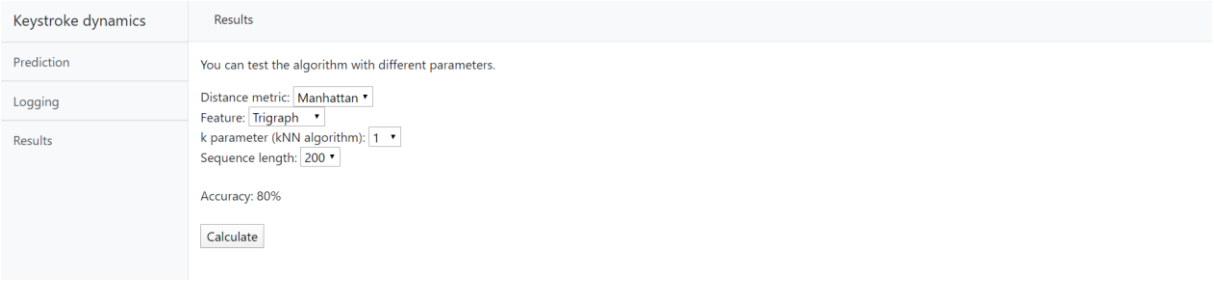
Algoritmus alkalmazása azonosításra



Amikor egy felhasználó gépel, akkor eseményeket generál. Ha összegyűlik egy előre meghatározott esemény szám, akkor megtörténik a felhasználó azonosítása. Ekkor a nyers esemény adatok elsőnek előfeldolgozásra kerülnek, így előállítódik jellemző attribútumok sorozata strandardizálva. Ennek a sorozatnak az elemeire egyenként lefut a k – legközelebbi algoritmus, ami mindegyik bemenetre tippel egy felhasználót. Így van egy sorozatom, aminek

az elemei felhasználók. Ennek a sorozatnak egyszerűen meghatározom a legtöbbet előfordult elemét, ami a végső eredmény. Ez az eredmény mondja meg, hogy kigépelte be a szöveget.

Webes alkalmazás



Egy webes alkalmazással valósítottam meg az adatgyűjtést és a tesztelést is. Az algoritmus futását egy python program végzi, ami AJAX kérésekkel érhető el a webes felület számára. A python 3.6.1 verzióját használtam és a k-legközelebbi szomszéd algoritmust, a távolság függvényeket és a standardizlás az sklearn könyvtárból importáltam. Az adatokat egy MySQL adatbázisban tárolom, kettő táblában. Az első tábla tárolja a felhasználókat, a második a felhasználók gépelését nyers formában.

Az alkalmazás három fő részből tevődik össze, az egyik ahol rögzíteni lehet a gépelését a felhasználónak, a másik, ahol azonosítani lehet valakit a gépelése és a korábban rögzített gépelése alapján és a harmadik felület, ahol az összes rögzített teszt adaton lehet az algoritmus teljesítményét mérni.

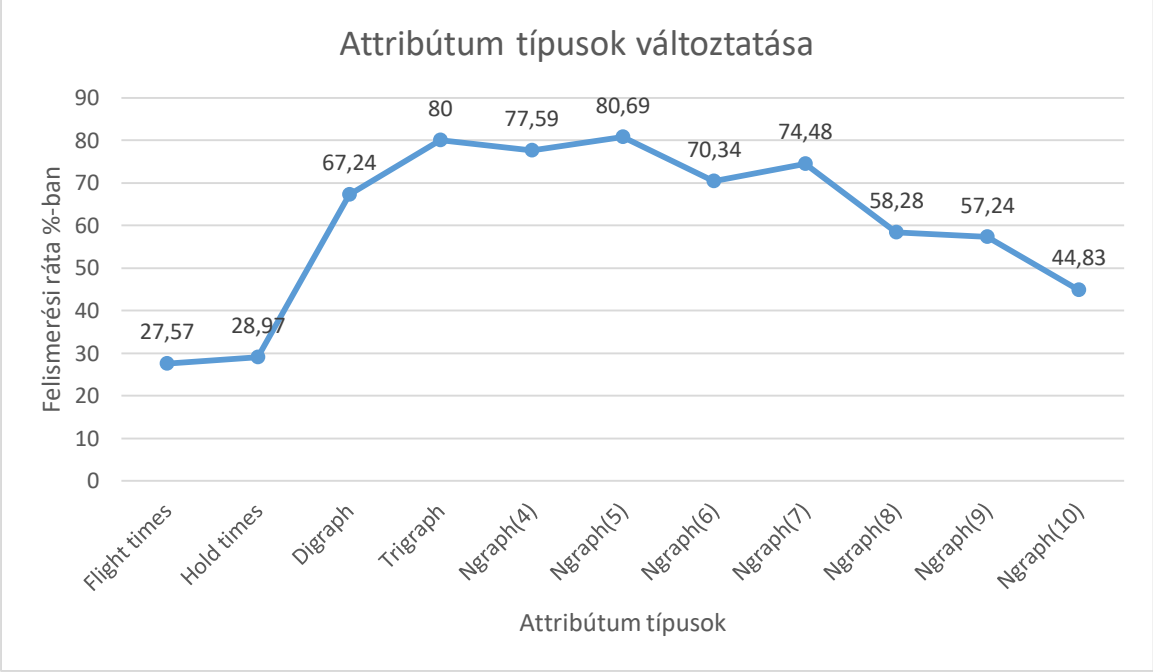
Eredmények

Ebben a fejezetben végig tesztelem az algoritmust különböző paraméterekkel és meghatározom az értékek egy olyan halmazát, amire a legjobb eredményt kapom. A legjobb eredmény alatt azt értem, amikor az algoritmus a legnagyobb arányban ismeri fel helyesen a felhasználókat.

A teszteléshez használt adatgyűjtésnél egy 1748 karakter hosszú angol szöveget kellett begépelnie 58 felhasználónak. Egy felhasználó gépelése 3500 körüli eseményt generált, ez a szám azért változó különböző személyeknél, mert az elgépelések és utána a javítások plusz eseményként jelennek meg az adatbázisban.

Az algoritmus „tanításához" az első 2500 eseményt használtam, míg a teszteléshez az utána következő maradék 1000 eseményt.

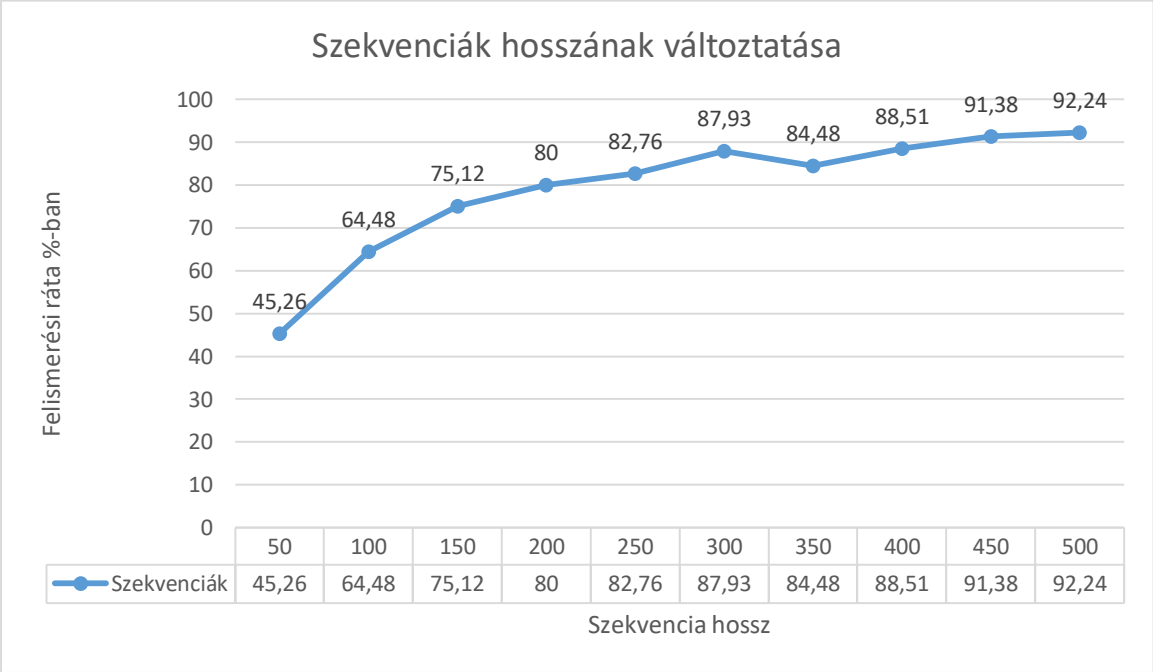
Elsőnek különböző attribútum típusokkal teszteltem az algoritmus teljesítményét. A többi paramétert alapértékeken hagytam. (k-paraméter: 1, távolság: euklideszi, szekvencia hossz: 200)



A távolságmérés függvényeknél kettő teszteltem, az euklideszi és a manhattan távolságot. A többi paramétert szintén itt is egy alapértéken hagytam. (k-paraméter: 1, attribútum: trigraph, szekvencia hossz: 200)

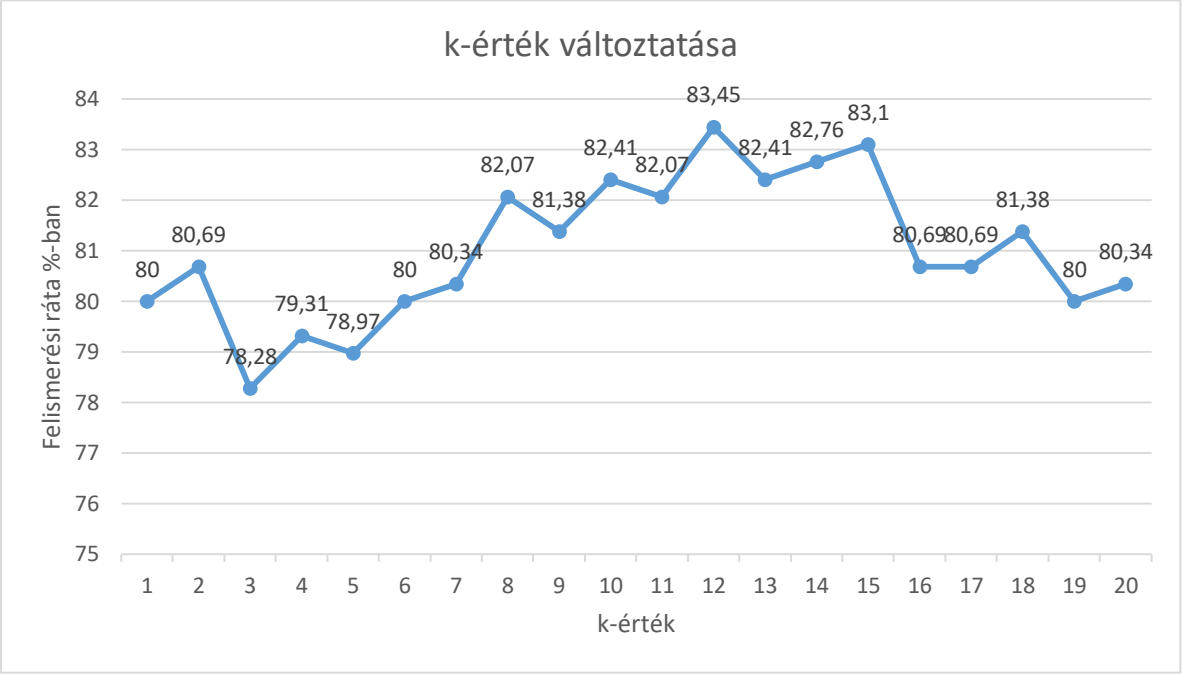
Távolság függvény:	Euklideszi	Manhattan
Felismerési ráta %-ban:	75,17%	80%

Következőnek a szekvencia hosszát változtattam. Itt egy szekvencia azt jelenti, hogy mennyi esemény utána fusson le az algoritmus és annyit is fog használni a felhasználó meghatározásához. Egy 50 hosszú esemény 25 karakternyi szöveget jelent szóközökkel együtt, ha közben nem történik elírás, majd törlés (backspace) és javítás események. Abban az esetben, ha történik elírás, törlés események, akkor azokat nem szűröm ki, hiszen azok is jellegzetessé tehetik egy felhasználó gépelési szokásait. A többi paraméter itt is alapértékeken. (távolság: euklideszi, k-paraméter: 1, attribútum: trigraph, szekvencia hossz: 200)



Itt lényegi különbség van, ahogy a szekvencia hosszát növelem, úgy az algoritmus felismerési képessége is egyre javul. Amit figyelembe kell venni, hogy ha túl hosszúnak választjuk a szekvencia hosszt, úgy az algoritmus futási sebessége romlik, illetve túl sokat kell begépelni ahhoz, hogy az események száma elérje az adott értéket. Jelen esetben az 500 hosszú szekvencia két hosszabb mondatnak felel meg.

Most a k-legközelebbi algoritmusnak a k paraméterét teszteltem k = 1 és k = 20 értékek között. A többi paramétert egy alapértéken hagyom. (Távolság: euklideszi, attribútum: trigraph, szekvencia hossz: 200)



Ebben az esetben nincs szignifikáns különbség az értékek között. A k = 12 esetben érte el a maximum értéket (83,45%), de k = 1 és k = 12 között sem szigorúan monoton növekvően ért el odáig, közben sokszor csökkent is. Majd k = 12 után csökkeni kezdett.

Összefoglalás

Összeségében sikerült elérni a kitűzött célt. Az algoritmus 92% pontossággal képes azonosítani a felhasználót az előzőleg rögzített 58 ember gépelése közül. A paramétereknél az algoritmus hatékonyságára a legnagyobb hatással az távolság függvény és a szekvencia hossz van. A k-paramétert érdemes 1-nek állítani és Trigraph attribútomokat használni, míg a szekvenci hossz 500-ra állítani és a manhattan távolságot használni.

A jövőben érdekes lehet kipróbálni gépi tanulással is az azonosítást.