**BOAW technika használatának eredményessége a természetes beszéd emóció alapú osztályozásánál**

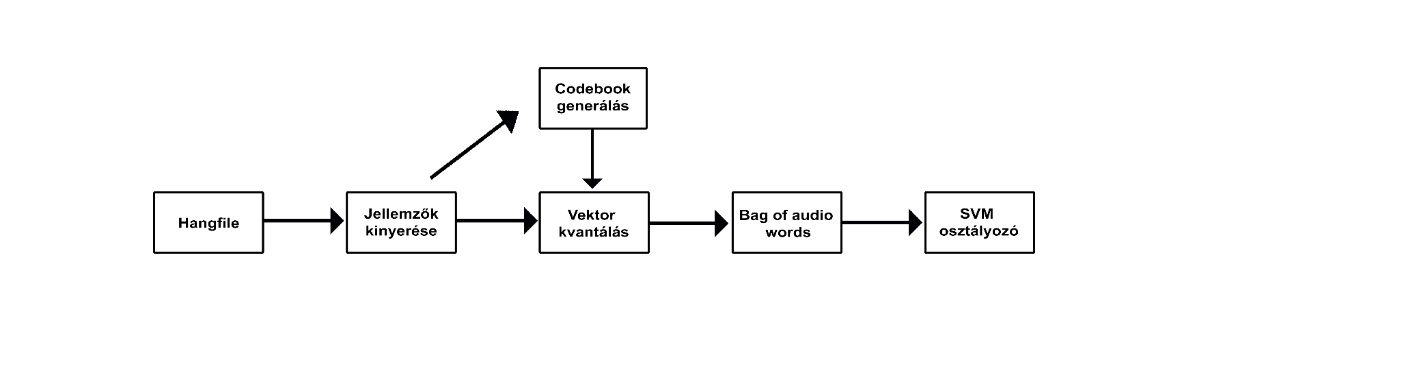
Napjainkban az automatikus érzelemdetektálás egy aktívan kutatott témakör. A gépek által használt érzelem-felismerő és -monitorozó rendszerek jelenleg is fejlődésben vannak. A technika alkalmazási köre elég széles skálán mozog. Többek közt hasznos az ember-gép interakciók során (az ember kommunikációjának monitorozására), a dialógus rendszereknél, az égészségi állapot felméréseknél, valamint a call-centerekben.

Jelen kutatás célja, hogy a BOAW technikát felhasználva jobb eredményt érjünk el a természetes beszéd emóció alapú osztályozásakor.

A kutatás során használt adatbázis 97 magyar anyanyelvű és magyarul beszélő személy hangját tartalmazza. A beszédek televíziós műsorok során lettek felvéve. A szegmensek túlnyomó része érzelmekben gazdag, folyamatos, spontán beszédből lett kivágva. Kisebb részük improvizációs szórakoztató műsorból jön. Ebből fakadóan az elsőként említett kategóriába tartozó minták a színészi játék miatt, az érzelmek egy feljavított és egyértelműbb változatát tartalmazzák, míg a maradék improvizációs halmazban lévők közelebb állnak a hétköznapi, természetes érzelemkifejezéshez. Az adatbázis összesen 1111 mondatot tartalmaz, melyek egy 831 elemű tanító és 280 elemű teszt halmazra lettek osztva. Az osztályozás során négyféle érzelmet definiálunk a beszédekben: Harag, Öröm, Szomorúság és Semleges hangulat.

Korábbi tanulmányok melyek ugyan ezzel az adatbázissal dolgoztak, 66-70%-os pontosságot tudtak elérni.

Az általunk használt technika, azaz bag of audio words hasonló a szövegfeldolgozásban ismert bag of words és a képfeldolgozásban alkalmazott bag of visual words módszerhez.



Első lépésben a hangfileokból kinyerjük az előre meghatározott jellemzőket, melyekből minden példához egy jellemzővektor áll elő. Egy korábbi kutatás alapján, most az alábbi 65 jellemzőből álló készletet használtunk.

!!!!!!

we consider the 65 acoustic low level descriptors (LLDs) and their first order derivatives (producing 130 LLDs in total) that were used for the INTERPSEECH Computational Paralinguistic challengE since its 2013 edition [42]. The COMPARE feature set has been computed with the open source extractor OPENSMILE (release 2.0) [43]. This feature set includes a group of 4 energy related LLDs, 55 spectral related LLDs, and 6 voicing related LLDs, cf. Table 1 and step 1 in Algorithm 1. For more details on the COMPARE feature set, the reader is referred to [44]. In what follows, we denote with Nt the temporal length of each speech sequence, with Nf the total number of acoustic features, with Ne the number of evaluators for each speech sequence, and with Nsp the number of speakers for which data and annotations are available as training material

table 3 from:

<https://diuf.unifr.ch/main/diva/recola/data/Pattern_Recognition_Letters_HCI_2014.pdf>

!!!!!!!!

Ezután a jellemzővektorokból klaszterizálás segítségével elkészül a codebook. A folyamat során megadott számú csoportot hozunk létre, ahol a klaszterek középpontjai lesznek a kódszavak. A csoportok számát nevezzük a codebook méretének. (-size, k-means VS kmeans ++).

A következő lépés a vektor kvantálás, mely során az eredeti jellemzőket kvantáljuk az előző lépésben generált kódszavaktól vett minimális Euclideszi távolság alapján. Az eredeti jellemzővektorok helyettesítésre kerülnek a hozzájuk legközelebb lévő kódszó indexével.

Végül egy hisztogramot készítünk a kódszavak és hozzájuk sorolt vektorok gyakoriságából. Ebből adódóan a hisztogram mérete megegyezik a codebook méretével. Az így előállított vektorhalmaz lesz a bag of words, ami a tanító algoritmusunk inputjául szolgál.

A tanítás során pythonban íródott SVM algoritmust használunk, melyet kezdetben 10 folds crossvalidationnel értékelünk ki. Tehát az aktuális mintahalmazt 10 egyenlő részre osztjuk és minden lehetséges 9 tanító - 1 tesztelő halmaz kombinációban (tanítunk a 9/10 részen és kiértékelünk az 1/10 részen) pontosságot számolunk. Az adott modell „jóságának” mérésére UAR-t (unweighted average recall: az adott osztályra helyesen osztályzott példák száma osztva az összes helyesen osztályozott példával) számítunk. Végezetül a 10 kombinációnál kapott UAR-ok átlaga adja a meghatározott mintahalmazra való tanulás képességét.

Ezen számok alapján valamint a 831 elemű tanító halmazunk használatának segítségével választottuk ki, hogy a kutatás egyes fázisaiban az openXBOW program mely paraméter beállításaival dolgozzunk tovább. Az utolsó körben legjobbnak ítélt kapcsolókkal legyártottuk a 280 elemű teszt halmazunkhoz tartozó bag-of-words jellemzővektorokat és crossvalidation nélkül ismét UAR-t számoltunk. Ezek az eredmények adták a tanulmány végső következtetésének vázát.

Tesztek:

Az openXBOW program kapcsolóinak próbái során a „-norm 1” kapcsoló végig alkalmazva volt, mellyel elértük, hogy a legenerált vektorhalmaz normalizálva lett. A programban implementált normalizálás a term frequency-ket osztja az inputok számával.

Az kapcsoló tesztelésének első próbájánál azt hasonlítottuk össze, hogy a codebook generálása során, milyen mintavételezési technikát érdemesebb használni. Két lehetséges metódust vettünk figyelembe, a „random”-ot és a „random++”-t.

A random egy teljesen véletlen mintavételezést biztosít, míg a random++ !!!!!! Generate the codebook by random sampling of the input feature vectors with weighting similiar to initialization of kmeans++. !!!!!!!

Az ..táblázatban látható eredmények alapján a további tesztekre a random++ kapcsolót állandósítottuk.

A következő összehasonlítandó tényezők: -normalizeInput, -standardizeInput, vagy ezek hiánya. A „-normalizeInput” segítségével a !!!!!! Normalize all output features (term frequencies, min->max is normalized to 0->1). !!!!!!

A „-standardizeInput”-tal elérhetjük, hogy !!!!! Standardize all output features (term frequencies).!!!!

A …táblázat alapján a két kapcsoló közel azonos teljesítményt nyújt, ám ezek hiánya romlást eredményez a tanulás minősége során, ezért a további teszteket normalizációval és standardizációval végeztük el.

Az utolsó összehasonlítási alapunkat a „-a” kapcsoló nyújtotta (mivel default értéke 1, ezért az eddigi teszteknél implicit módon szerepelt). Ennek segítésével megszabhatjuk, hogy a codebook generálása során, !!!! When creating the bag-of-words, assign each input feature vector to p closest words from the codebook. (default: a=1, only closest word)

In case of several codebooks (see -attributes), a different number can be specified for each codebook using separator comma, e.g., -a 5,2)!!!!

Háromféle különböző beállítását néztük meg: -a 1, -a 5 és -a 10. A …táblázatban látható eredmények jól mutatják, hogy az 5 és 10 legközelebbi szót nézve közel azonos javulást kapunk, így ezekkel végeztük el a végső kiértékelést.

Az eddigi döntéseket a tanító halmazon végzett crossvalidation technika által adott százalékok alapján hoztuk meg. Ezzel határoztuk meg, azt a szűkebb paraméterhalmazt, melyere a teszt mintákon is kiértékeljük az SVM algoritmust. A train halmazzal és az egyes openXBOW kapcsolókkal való új jellemzők generálás során az elkészült codebook-ot minden esetben lementettük, így ugyan ezt használtuk az adott beállításhoz tartozó, teszt halmazra számolt jellemzők generálásánál.

Minden eddigi vizsgálatot a pontosabb kiértékelés érdekében 11 különböző codebook mérettel végeztünk el. (32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768) A …diagramon látható eredmények azt mutatják, hogy a korábbi tesztek alapján a teszt halmazon való kiértékelést elegendő csupán az 1024 feletti méretekre elvégezni, ugyanis azalatt a tanulás konzisztensen rosszabbnak bizonyul.

Teszt halmazon kapott eredmények összegzése:

Codebook size:

When generating a small set of clusters, dissimilar sounds (as modeled by the MFCC vector) can be grouped into the same audio-word, and the codebook is therefore more general but not necessarily discriminative. On the other hand, larger codebook sizes assign similar sounds to different audio-words, resulting in a more discriminative but less general vocabulary. Computation time to generate the codewords also increases with larger codebook sizes.