SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR
GÉPÉSZMÉRNÖKI, INFORMATIKAI ÉS VILLAMOSMÉRNÖKI KAR

DIPLOMAMUNKA

Gépi tanulási módszerek szövegfeldolgozáshoz

Fazekas Zoltán

Villamosmérnök MSc
Automatizálási szakirány

2022

Nyilatkozat

Alulírott, **Fazekas Zoltán (CWGK9I)**, Villamosmérnök MSc szakos hallgató kijelentem, hogy a *Gépi tanulási módszerek szövegfeldolgozáshoz* című szakdolgozat feladat kidolgozása a saját munkám, abban csak a megjelölt forrásokat, és a megjelölt mértékben használtam fel, az idézés szabályainak megfelelően, a hivatkozások pontos megjelölésével.

Eredményeim saját munkán, számításokon, kutatáson, valós méréseken alapulnak, és a legjobb tudásom szerint hitelesek.

Győr, 2022.12.15

_____

Fazekas Zoltán

Kivonat

Gépi tanulási módszerek szövegfeldolgozáshoz

A beszédfelismerés azon feladat, mely során a beszédfelismerést végző rendszer azonosítja a kiejtett beszédjeleket és átalakítja ezeket szöveggé, vagy más, számítógép által feldolgozható adattá.

A beszédfelismerő alkalmazások segítségével a különböző berendezések képesek értelmezni az emberi beszédet. A beszédfelismerés és a hangfelismerés fogalmát sokszor összekeverik, pedig a beszédfelismerés a beszélt nyelv értelmezését jelenti, míg a hangfelismerés a beszélő személy azonosítását, annak hangja alapján.

Az emberhez hasonlóan a felismerő rendszernek is szüksége van tanulásra. Mind a nyelvi, mind az akusztikus információkat meg kell tanítani a rendszernek, a használat előtt. A felismerés célja és a rendelkezésre álló erőforrások döntik el a használandó rendszert, de manapság ez már szinte kizárólag mesterséges intelligencia segítségével történik. Ennek fontos része a *deep learning* vagy *mély tanulás*, amely elsősorban a többrétegű visszacsatolt rendszerek segítségével történik. A dolgozatban először az elméleti alapokat mutatom be, a széles matematikai háttér főbb elemeivel. Szóba kerülnek a különböző felismerő módszerek, modellek is.

Abstract

Machine learning methods for text processing

The speech recognition is a task, when the specific system recognize the spoken language pieces (phonemes) and transform this to format, which usable for computer for processing.

With the usage of speech recognition applications, the different systems can understand the human speech and react, if need. The speech recognition and speaker recognition are often mixed. The speech recognition is the understanding of the spoken language, why speaker recognition aim is to identify the speaker, based on the voice.

As human persons, than these systems also require trainings. The language and acoustic information need to feed for the systems before usage. The best system is selected for the specific task, based on target of recognition and the available resources. Nowadays are these system exclusively based on Artificial Intelligence/Machine Learning. Important part of the solution is the deep learning, which mainly represented with multilayer recurrent neural networks.

In this work I present first the theoretical bases, with the main elements of the well defined mathematical background. I show and analyze more recognition systems and NLP models.

Tartalomjegyzék

1. Bevezetés

Mára a mesterséges intelligencia (**Artifical Intelligence, AI**) és az ehhez kapcsolódó gépi tanulási megoldások (**Machine Learning, ML**) a mindennapunk részévé váltak, látható (TELEKOM chat alkalmazás [1]) vagy kevésbé látható módon (Google reklámajánló rendszer [2]). Ezek a megoldások egyre fejlettebbek és egyre szélesebb kör számára hozzáférhetőek.

A járműipar is számos területen használja ezeket a technológiákat. A gyártási folyamatokat segítő rendszereken, a logisztika okos kezelését biztosító programokon, raktárrendszereken felül a termékekbe is beépültek. Az útvonaltervezés, a közlekedési táblák felismerése, a vezetővel történő kétirányú kommunikáció, mind-mind használják az AI és ML vívmányait. Az alkalmazások egy része a jármű rendszerén belül fut – pl. a szabályzást segítő szoftverek -, míg mások a mobil hálózatokon keresztül intenzíven használják a felhőalapú megoldásokat – pl. útvonaltervezés, optimalizálás, komplex beszédfelismerés.

Jelen dolgozat témája a beszédfelismerés ML alapú rendszerekkel és annak járműipari felhasználhatósága. Az alapok rövid megismertetése után körbejárom a rendelkezésre álló megoldásokat, rámutatva azok előnyeire, hátrányaira, kitérve a járműipari értékekre vagy esetleges nehézségekre.

A dolgozat utolsó részében az elmélet gyakorlati oldalról való ismertetése található.

2. Gépi tanulás – Machine Learning

Ebben a fejezetben röviden bemutatom a gépi tanulás kialakulását és jelenlegi főbb irányzatait.

Mit is értünk ML alatt? Nem mindent, amit számítógép végez. Mitchell definíciója szerint: *„Azt mondjuk, hogy egy számítógépes program tanul E tapasztalatból a T feladatok osztályára nézve P teljesítménnyel, ha a teljesítménye a T-beli feladatokon P-ben mérve javul az E tapasztalat megszerzésével."* [3]

A gépi tanulással szeretnék olyan bonyolult feladatokat megoldani, amelyeket klasszikus algoritmikus programozással nem lehet, vagy túlontúl időigényes. Olyan feladatok esetén, ahol nem pontos eredmény, hanem statisztikai valószínűség is megfelel a lehetséges kimenetekhez szintén szóba jön az ML. A bonyolult feladatokat fázisokra kell bontanunk ehhez, ami azt az előnyt is hordozza, hogy megfelelő tervezés esetén lehetőség van a különböző ML módszerek használatára, a legjobb eredmény elérése érdekében.

A kiindulási adatainkat sokszor tulajdonságok gyűjteményeként (**features**) írjuk le. Az eredményt annak pontosságával (**accuracy**) vagy a hibaaránnyal (**error rate**) jellemezzük. A módszerek fejlesztésénél az előbbi javítása, míg a második csökkentése a cél.

Attól függően, hogy milyen tapasztalatokat használunk fel vagy nem a tanulási folyamat során beszélünk felügyelt (**supervised**) és nem felügyelt (**unsupervised**) gépi tanulásról. Fontos itt megemlíteni, hogy a legtöbb mai rendszerben ezek kombinációja használt, ahol fázisonként, a fázis jellemzőinek megfelelő hol az egyik, hol a másik módszer használt.

2.1. A kezdetek

A gépi tanulás és a mesterséges intelligencia (**Artificial Intelligence**) kezdeteinek megismeréséhez az egyik első, ebben a témában megjelent hazai könyvből merítettem [4]. A történet 1956 nyarán kezdődött a Dartmouth College-ban, ahol *tíz vezető tudós hivatalosan is deklarálta az MI megalapítását.* [4, p. 2]. Az első korszak 10 évében nagyratörő tervek lettek megfogalmazva (pl. univerzális gépi fordítás), de a várakozásokhoz képest szerény eredmények születtek, főként a számítástechnika kezdetleges állapota miatt. Kiemelendő, hogy ebben az időszakban sok olyan elmélet keletkezett – magának a tudományágnak az alapjai mellett - ami későbbi fejlesztések alapját adta, és a játékelméleti felhasználások voltak ezidőben többségben.

A 70-es évek tágabb időszaka a korábbi kudarcok alapján már adott, szűkebb feladatok kidolgozására koncentrált. Ilyenek voltak pl. az általános fordítás helyett szakterületekre koncentráló fordítási próbálkozások.

A harmadik korszak a 70-es évek végén vette kezdetét. Ahogy a korábban már idézett könyv leírja, *a MI konzekvens módon tovább vitte a megkezdett szemléletváltást és az ismerettechnológia felé fordult.* [4, p. 3]. A gépi beszédfelismerés is jól szimbolizálja ezt az irányt, ahol a speciális megoldások teszik lehetővé – az azóta nagyságrendekkel nagyobb számítástechnikai kapacitás, az internet vagy a felső alapó szolgáltatásokon túl – hogy a mindennapi élet különböző kihívásaira a lehető legjobb megoldások szülessenek.

2.2 A gépi tanulás napjainkban

A mesterséges intelligencia azon kutatási ágát, melyben a modellt minta adatok felhasználásával, szabályok explicit programozása nélkül állítja elő gépi tanulásnak (ML) nevezik [5, p. 29]. A rendszer a tanító adathalmazban rejlő minták alapján próbál szabályszerűségeket keresni. Amennyiben sikeres a tanítás, általános helyes választ ad a rendszer a problémára.

A tanításhoz felhasznált adatok jellegük szerint két típusra bonthatók:

1) tipikus bemeneti minta adatok és azok modellezendő, helyes megoldása (címke) is az adathalmaz része

2) címkézetlen bemeneti paraméterek halmaza. [5, p. 29]

Megjegyzendő, hogy ezen cimkék előállítása vagy gépi algoritmusokkal (**classification**) vagy emberi erőforrással, az adatok manuális megjelölésével történhet. A tanítási feladat jellege és a tanítási adatok típusa szerint három nagyobb kategóriába sorolhatók az ML módszerek

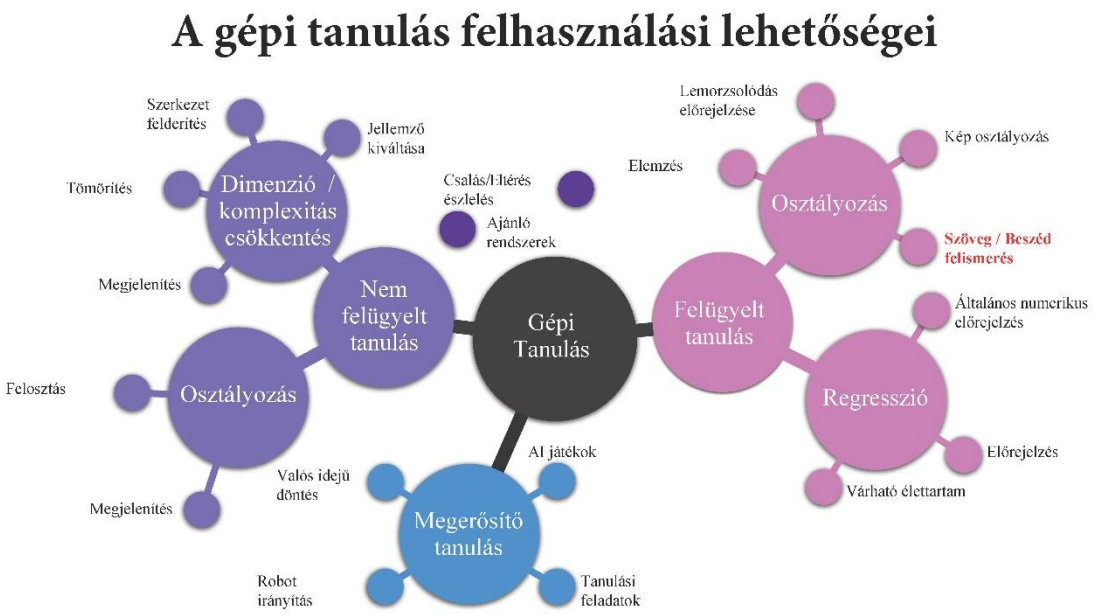
- A felügyelt tanítás (**supervised learning**) előre címkézett adatok segítségével történik. A tanítás célja a bemenő objektum lehető legpontosabb leképezése a kimeneti információra.
- A felügyelet nélküli tanítás (**unsupervised learning**) során a tanító halmaz nem tartalmaz címkéket, a módszer célja a rendszer belső struktúrájának feltérképezése bármilyen prekoncepció nélkül. A probléma felfogható mint dimenzióredukciós feladat.
- A tapasztalatok alapján zajlik a megerősítő tanulás (**reinforcement learning**), melynek feladata egy célfüggvény maximalizálása. Ennek érdekében a program dinamikusan, környezetéből nyert visszacsatolás révén hangolja modelljét. Ez a módszer nagyban hasonlít a gyerekkori tanulás módjára, ahol sok ismeretet a környezet visszajelzése, a pozitív vagy negatív tapasztalatok megerősítése révén alapján történik.

A három alaptípuson kívül napjainkra sok egyéb módszer is kialakult, ebből talán kiemelkedik a gépi tanulás területét forradalmasító mesterséges neurális háló (**Artifical Neural Network**) tekinthető, mely a biológiai neuronok alapszintű viselkedését próbálja a lehető legegyszerűbb matematikai eszközökkel szimulálni. A gyakorlatban egy sor különböző, adott típusú problémákra optimált mesterséges neurális modellt alkottak, amelyekből több a gépi beszédfelismerés legfőbb alkotó eleme.

A módszer fontosságát illusztrálja az is, hogy mind a lineáris regresszió, mind szinguláris érték felbontás, mind a variációs Monte-Carlo módszer – a 3 alaptípus jellemző módszereit - tekinthető egy-egy mesterséges neurális háló egyszerűsített határesetének (lineáris regresszió: zérus rejtett rétegű neurális háló; szinguláris érték felbontás: lineáris autoencoder; variációs Monte-Carlo módszer: Boltzmann-gép).

3. Mély tanulás – Deep Learning

A tanulási / tanítási folyamat az elsődleges lépése a beszédfelismerésnek. Ebben a fejezetben a Deep Learning és a Natural Language Processing (NLP) módszereit ismertetem. Ezek jobb megismeréséhez szükséges tisztázni több alapfogalmat. Az *1. ábra* segít összefoglalni a főbb felhasználási területeket.



1. ábra. A gépi tanulás egyféle osztályozása.

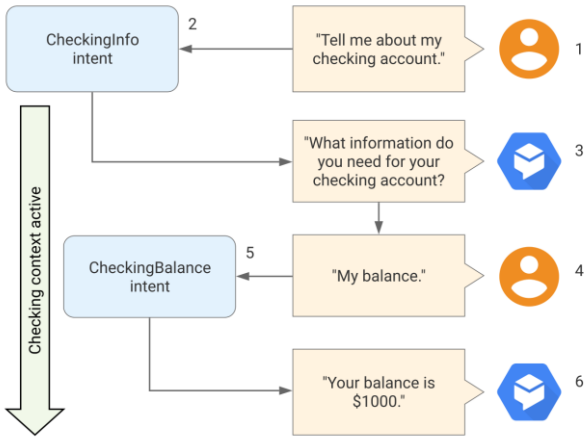
3.1 Alapfogalmak

3.1 definíció: „*Nyelvnek tekintem a mondatok valamely (véges vagy végtelen) halmazát; minden egyes mondat véges hosszúságú és megadott elemek véges halmazából épül fel. [...] Valamely Ny nyelv nyelvészeti elemzésének alapvető célja az, hogy a nyelvtanilag helyes sorozatokat, amelyek Ny mondatai, különválasszuk a nyelvtanilag helytelen sorozatoktól, amelyek Ny-nek nem mondatai. [...] Ny nyelvtana ily módon olyan motor lesz, amely Ny valamennyi nyelvtanilag helyes sorozatát létrehozza, azaz generálja, de nem generál egyetlen nyelvtanilag helytelent sem.* [6, p. 11]”

Az NLP elsődleges feladata ezen a hétköznapi értelemben értett természetes nyelvek felismerése és kezelése. A tudomány definiált úgynevezett formális nyelveket, amelyek matematikai módszerekkel, ellentmondás nélkül leírhatók. Ezek közé természetesen nem illeszthetők a köznapi nyelvek – és főként nem a különböző regionális nyelvek vagy dialektusok – de segítenek az NLP használatához szükséges modellek megalkotásában.

A nyelvek nagy száma és komplexitása miatt nincs mód az összes nyelv, minden mondatát egy kérdés-válasz felépítésű struktúrába rendezni. Ha ez még sikerülne is, a szavak vagy mondatok absztrakt, elvont jelentésének problémáját megközelíteni sem lehetne.

3.2 definíció: *„Kontextus alatt azon körülmények és állapotok összességét értjük, melyek egy vagy több eseményt absztrakt szinten összekapcsolnak, körülölelnek.* [6, p. 11]”



2. ábra. A kontextus kezelés egy lehetséges megoldása [7].

A *2. ábra* független attól, hogy írott kommunikáció – pl. chatbot – vagy beszéd alapú társalgás folyik a felhasználó és a rendszer között. Az eltérés csak a szöveg/beszéd felismerésének illesztésében, az interface-ban van. A kontextusok között mindig hierarchia van. A beszédfelismerés

során ezt az elvárható, prediktív hierarchiát ki is használhatjuk és a legtöbb NLP rendszer ezt meg is teszi.

3.3 definíció: *„Entitásnak nevezünk valamit, ami más dolgoktól függetlenül létezik, megvan a maga független létezése* [6, p. 14]"

Minden olyan objektumot entitásként kezelünk, amely elkülöníthető a többitől és a tulajdonságai jól meghatározhatóak. Mivel az ML algoritmusok, a tanulási folyamat matematikai alapokra épül, igy az első lépés mindig ezen entitások megfeleltetése numerikus azonosítóknak. Ez az átalakítás lehetővé teszi a különböző ML/DL módszerek használatát, ezek kombinációját és a megfelelő feldolgozási sebességet is. Ezek a megfeleltetések valójában adatbázisokat jelentenek, de gyakran ezek nem statikus adat összerendelések, hanem a tanítási folyamat során dinamikusan összerendelt párosítások – pl. különböző személyek hangfelismerésének kezelése.

3.4. definíció: *„A beszélgetés célja nem más, mint egy várt eredmény, amelyet terveznek, vagy amely irányítja a tervezett tevékenységeket.* [6, p. 15]"

A sima beszédfelismerési feladatoknál ez a cél azt a pontot jelenti, amikor a felismerő rendszer nagy valószínűséggel – emlékezzünk arra, hogy az ML alapja statisztika – meg tudja határozni, hogy

- Mi a szöveg tartalma?
- Mi az a cél, amit a beszélő el szeretne érni?
- Milyen választ – pl. további kérdéseket – vagy akciót kell adni?

Ahogy minden beszélgetésünknek, igy a beszédfelismerési feladatoknak is mindig van valami absztrakt, elvont célja, mely felé próbáljuk terelni a másik felet, akivel a beszélgetés zajlik. Az NLP algoritmus célja, hogy a programozott példák, az ismert entitások, valamint a megadott kontextusok alapján eldönti, hogy a felhasználó mit szeretne.

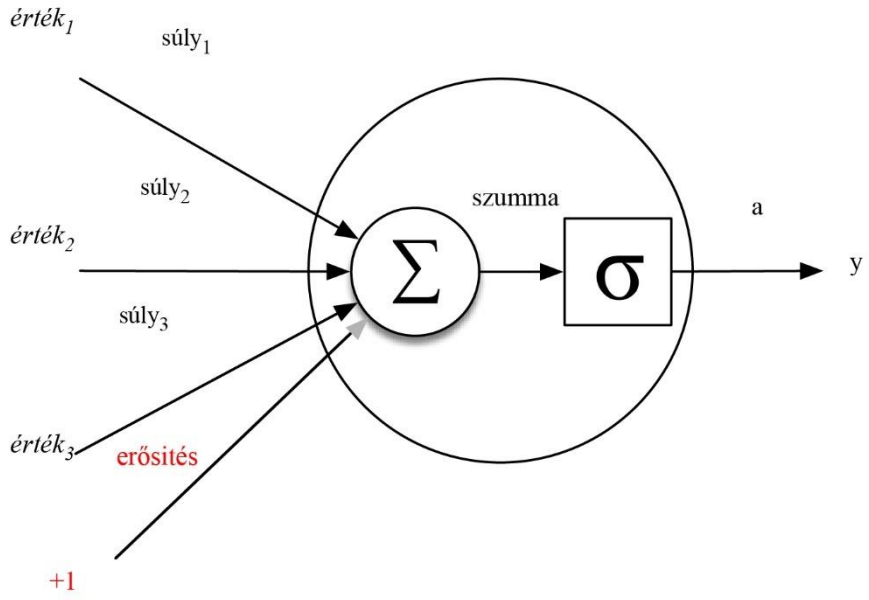
Természetesen az NLP célja az is lehet, hogy a felhasználót – a megfelelő kérdések feltételével és a válaszok korrekt osztályozásával – egy, a program által meghatározott irányba terelje, befolyásolja. Pl. pizzát válasszon ebédre, ne hamburgert. Ezek a rendszerek azt használják ki, hogy a többség a verbális információ útján könnyebben meggyőzhető, mint egyszerű hirdetés útján [7].

3.2 Neurális hálózatok

A neurális hálózat egy párhuzamosan elosztott feldolgozó rendszer, amely, egyszerű feldolgozó elemi egységekből épül fel. Ezek legfőbb tulajdonsága a korábbi tapasztalásokból szerzett tudás eltárolása és más rendszerek számára hozzáférhetővé tétele. A biológiai idegrendszer működése alapján definiálták a neurális hálózatokat. Ezek két jelentős szempontból hasonlítanak az agyra:

- a tudást a hálózat tanulás során a környezetből szerzi meg,
- a súlyok tárolják a megszerzett tudást. [8, p. 4]

Ezt úgy is megfogalmazhatjuk, hogy a mesterséges neurális hálózatok az emberi agy egyféle modellezési fajtái, ahol a feldolgozó egységek neuronokként működnek. Egy neuron modelljét a *3. ábra* alapján érthetjük meg.

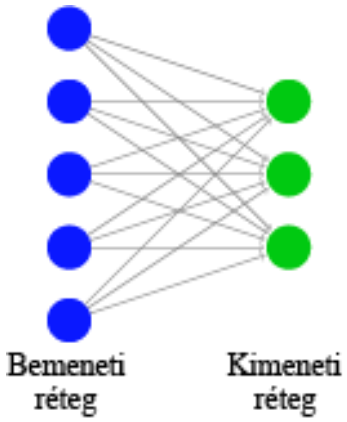


3. ábra. A neurális hálózat egy elemének sematikus ábrája.

Az elem 3 bemenetet kezel, azok értékeit a beállítható súlyozásokat összesítse. Ezt még módosítani lehet az *erősítés* értékkel. A kimeneti érték az *y*. Az érthetőségért bevezetünk néhány köztes változót. Ilyen az összegzés eredménye (szumma) és az *a* ami a *σ* művelet utáni eredmény. A két eredmény nem ekvivalens és a mély tanulási feladatoknál mindkettőre szükség lehet.

3.2.1. Egyrétegű előre csatolt neurális hálózat

Az egyrétegű modell (**single-layer feedforward network**) egy olyan hálózatot épít fel, ahol minden bemeneti egység kapcsolatban áll minden kimeneti egységgel. A kimeneti réteg csúcsai nem csatlakoznak a bemeneti réteghez, ezért előrecsatolt hálózat – a *4. ábrán* csak egyirányúak a nyilak. A bemeneti csúcsok nem végeznek műveletet, ezért egyrétegű a modell.



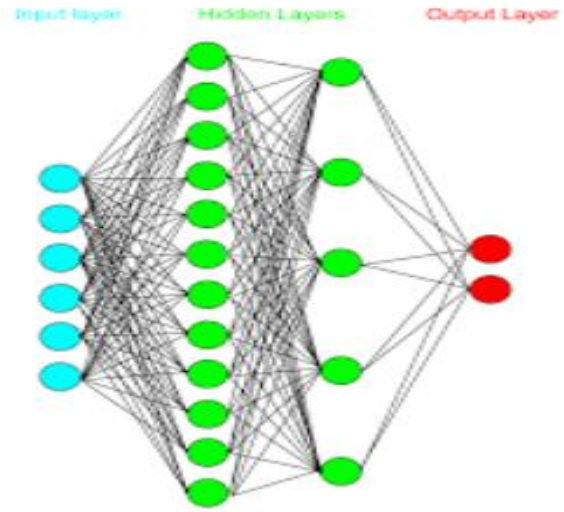
4. ábra. Egyrétegű előre csatolt hálózat sémája.

3.2.2. Többrétegű előrecsatolt neurális hálózat

A többrétegű (**multilayer**) előrecsatolt hálózatok tartalmaznak egy vagy több rejtett (**hidden**) réteget, amelyek szintén számítási egységek (zöld csomópontok a lenti ábrán).

A rejtett elnevezés eredete az, hogy a hálózat ezen részeit nem látjuk közvetlenül a tanulási folyamat során. Ezen rétegek hozzáadásával a hálózat képessé válhat magasabb rendű statisztikai eredményeket kinyerni a bemeneti értékekből. Így globálisabb képet kaphatunk. A hálózatban a bemeneti réteg az első rejtett réteg neuronjaihoz kapcsolódnak. Végül az utolsó rejtett réteg a kimeneti réteghez kapcsolódik. A kimeneti rétegből származó jeleket tekintjük a neurális hálózat válaszának a bemeneti kombinációra vonatkozóan.

Az *5. ábra* egy ilyen, többrétegű előrecsatolt neurális hálózatot ábrázol [9, p. 19].



5. ábra. Többrétegű neurális hálózat sémája.

3.2.3. Visszacsatolt neurális hálózatok

A visszacsatolt (**recurrent**) neurális hálózat – amely az NLP és a beszédfelismerési feladatok legfontosabb megoldási módja - abban tér el a korábban bemutatott modellektől, hogy van benne visszacsatolást végző hurok. Más szavakkal van olyan kapcsolat a hálóban, ami egy későbbi – általában rejtett - rétegből, egy korábbiba vezet, ami lehet a bemeneti réteg vagy másik rejtett réteg. A visszacsatolásokat egyfajta memóriaként használja a neurális hálózat a tanítás és a működés során. Nem jellemző, hogy a kimeneti rétegből végzünk visszacsatolást, ennek nincs szerepe itt.

Az előző ábrát felhasználva ez úgy képzelhető el, hogy a zöld rétegek valamely neuronjából egy nyíl vezet az alatta lévő másik zöld vagy a kiinduló kék rétegbe. Elméletben nincs limit, sem a visszacsatolás mélysége, sem a visszacsatolások száma alapján – és ezek kombinációban -, de praktikus megfontolások – pl. a modell kezelhetősége, vagy a számítási kapacitás – limitálják ezt.

3.2.4. Konvolúciós neurális hálózatok

Bár a konvolúciós neurális hálózatokat (**convolutional neural networks**) képeken való mintafelismeréshez fejlesztették ki, az jól használható pl. szöveg klasszifikáció során is. A fő célja ennek a módszernek, hogy a kezeléshez szükséges neuronok számát jelentősen csökkentse a többihez képest, mivel pl. a rétegek számának növelésével nem lehet áthidalni a nagy méret problémáját.

Ennek az egyik oka a feladatokhoz képest még mindig korlátozottnak tekinthető számítási képesség, mivel minél bonyolultabb egy hálózat, a tanítás is annál költségesebb lesz. Hiába javul a számítási kapacitás, az adatbázisok növekedésével, a pontosabb modellekkel ez egy folyamatos verseny. A másik fontos ok a túltanulás (**overfitting**), amikor az általános összefüggések megtanulása helyett a modell, a minta egyedi sajátosságait rögzíti [10] [11].

A konvolúciós hálók esetén 3 réteget különböztethetünk meg:

- *konvolúciós*: itt tanítható szűrőket (**kernel**) használunk. Ezek a szűrők általában alacsony térbeli dimenzióval rendelkeznek, de őrzik a bemenet mélységének komplexitását.
- *összevonó* (**pooling**): kizárólagos célja a dimenziószám csökkentés. Ezzel a számítási feladatok összetettsége kisebb lesz és a túltanulást esélyét is korlátozza.
- *teljesen összekötött* (**fully-connected**): itt az összes neuron közvetlenül kapcsolódik a két szomszédos réteg összes neuronjához, de a rétegen belül nincsenek kötések.

4. A természetes beszédfelismerés alapjai és céljai

A beszédfelismerő modellek és az erre épülő rendszerek célja, hogy a gépen futó szoftverek megértsék, amit az emberek mondanak. Általában az elhangzott beszédet a gép számára értelmezhető formátumra, pl. írásos szöveggé alakítják ezek a módszerek. Ezen átalakításnak az elsődleges célja az adatok szűrése, és matematikailag leírható, értelmezhető formátumba hozása, vagy köznapilag nevezve digitalizálása.

Ezután valahogyan értelmezik és felhasználják a hangokból kinyert tartalmat. Ilyen feladatok lehetnek például a beszéd vagy a filmek feliratozása, de pl. TV műsorok valós idejű (**realtime**) alá-feliratozása hallássérültek számára. Ez utóbbi komoly kihívás, a különböző hangok, a gyors feliratozás igénye és a minimális hibázási lehetőség miatt. De olyan rendszer is elképzelhető, amely vezetés közben képes választ adni egy megadott kérdésre vonatkozóan, vagy akár beavatkozást végez – és ebben az esetben sem megengedhető a tévedés. A rekurrens és a konvolúciós neurális hálók alkalmazása gyakori ezekben a feladat típusokban [12] [13].

Az alábbiakban ismertetem a [14] publikációban bemutatott modellt.

A modellt arra tesztelték, hogy képes-e a megfelelő fonémákat társítani a hangokhoz. Ez az eljárás egy hibrid felépítésű módszer, amely ötvözi a kapcsolatos időzített klasszifikációt (**connectionist temporal classification**) és a konvolúciós neurális hálót. Vannak más módszerek is, amelyek konvolúciós hálózatok helyett általában visszacsatolt hálózatokkal dolgoznak, mivel jó eredményeket adnak a beszédfelismerés során. Ezek hátránya azonban a lassabb tanulás, és a gradiens értékek, igen nagyra nőhetnek.

Ahhoz, hogy az eljárás jól működhessen, szükség van hosszútávú feltételek használatára, emiatt több egymásba ágyazott konvolúciós réteget kell, hogy tartalmazzon a modell. Kisméretű szűrővel dolgozik, ami a helyi sajátosságok megtanulását segíti elő. A rétegek a következőképpen követik egymást:

- Bemeneti réteg
- Konvolúciós réteg, egy maxout aktivációs függvénnyel
- A maximumokat összevonó réteg
- További konvolúciós rétegek, szintén maxout-tal. Ez kilencszer ismétlődik meg
- 3 teljesen összekötött réteg, megint csak maxout aktivációs függvénnyel
- A végső kapcsolatos időzített osztályozás

4.1. Nyelvi modellezés területei

A nyelvi modellezés már a mindennapi életünkben is megtalálható, sokszor nehezen felismerhető módon. Az egyszerűbb szövegszerkesztőknél vagy a mobilokban is megtalálható helyesírás ellenőrzés, amely sokszor már stilisztikai tanácsokat is ad, vagy az automatikus kiegészítés, mind az NLP egy-egy felhasználása. Néhányat ismertetek az alábbiakban.

A *szöveg klasszifikáció* egy szerteágazó terület, ahol a vizsgált anyagokhoz cimkéket rendelünk, a megszabott feladat függvényében. Ilyen lehet pl az érzelem elemzés (**sentiment analysis**) – amely gyakran használt pl. vélemények relevanciájának elemzésekor -, de a *levélszemét szűrés* (**spam detection**), amely ma már egy közepes cég esetében is milliószámra szűri ki a kéretlen bejövő leveleket. Maga a *nyelv* vagy *műfaj felismerés* is klasszifikációs feladat.

A *gépi fordítás* a különböző nyelvek között próbál szöveget vagy beszédet fordítani egy adott nyelvről egy másik nyelvre. Erre jellemző példa a Google fordítója [15]. amely statisztikai alapú gépi fordítást (**Statistical Machine Translation, SMT**) használ [16]. Sok gépi fordítóprogram esetében, amely nemcsak 2 nyelv között akar fordítani, hogy nem direktben fordít két tetszőleges nyelv között, hanem egy közbenső nyelv felhasználásával, két lépcsőben végi a fordítást. Különböző neurális háló modelleket is kifejlesztettek a fordítóprogramok fejlesztésére [17], ez egyik lehetőség a visszacsatolt neurális háló használatára épül, pl kódoló-dekódoló (encoder - decoder) eljárással [18].

A *képszöveg generálás* egy olyan eljárás, amelynek célja, hogy a gépezet le tudja írni, hogy mi látható egy képen, vagyis egy rövid összegzést szeretnénk a képről kapni. Ez használható akár videó leírások, ajánlások készítésére is. Erre a feladatra sokszor használnak speciális visszacsatolt neurális hálókat.

A *dokumentum összefoglalás* során szeretnénk egy rövid összefoglalást kapni a megadott tartalomról, a szöveg adott szakaszáról. Ez lehet egy absztrakt készítése egy cikk bevezetéséhez vagy címadás a dokumentumnak. Kétféle módszer használatos: a modell használhatja csak a szövegből kinyert információkat, vagy korábbi tapasztalatokat is felhasználhat a végrehajtás során.

A *kérdés-válaszs* (**question answering**) típusú feladatok lényege, hogy egy meghatározott témakörből kell a gépnek kérdésekre válaszolnia úgy, hogy először feldolgozza a számára megadott, ide tartozó irodalmat [19]. A tanuló rendszernek meg kell találnia a kapcsolatot a kérdés és a szöveg között, ami egyfajta tartalmi értelmezést jelent. A kérdésekre – a használhatóság érdekében - általában egyszerű kifejezéseket várunk válaszként.

4.2. A beszéd felismerése

A klasszikus megvalósítása ennek a *Speech-To-Text* vagyis a beszédet írott szöveggé alakító konverzió [20]. Ez a legáltalánosabb módszer, mivel függetleníthető a beszélő személyétől, bár feltételezi, hogy a nyelv maga ismert, vagy a nyelv felismerését is be kell illeszteni a célfeladatok közé.

Az emberi beszéd – de az éneklés is - akusztikus hullámok formájában, a beszédhangok kibocsátásán alapszik. A gépi beszédfelismerés szintjén ez a beszédjelek (fonémák) elektronikus képe. A pontosabb meghatározás szerint a beszéd nem csupán fonémák sorozata, de fontos a hangsúlyozás, hanglejtés stb is.

A beszélők közötti szembetűnő különbség – köznapi életben is -, hogy mindenki másképp ejti ki a hangokat. Ez segít pl abban, hogy akár a csecsemők is felismerjük anyjuk hangját. Sőt kisérletekkel igazolták, hogy ugyanaz a személy sem képes kétszer pontosan ugyanazt a hangot produkálni – talán a színészek, énekesek képesek erre, akik jobban odafigyelnek a beszédükre és képzettek ezen a területen. Azok az emberek, akik bármilyen beszédhibával rendelkeznek, de pl a tájszólással rendelkezők esetében is tovább bonyolódik a gépi beszédfelismerés mikéntje. Ezen problémák többsége a lényeg kiemelés módszerével javítható. Itt az egyik lehetséges eljárás a lineáris predikció, vagy a Fourier transzformáció. Ha zajos a környezet, matematikai zajszűrő algoritmusok használatával javítható a felismerés, illetve több hangrögzítő eszköz egyidejű alkalmazása is javíthat a minőségen, ami elengedhetetlen a pontos felismeréshez.

Minden hangutasítással működő rendszernek a hatékony – sebesség és pontosság terén - beszédfelismerés a kritikus része és az autóipar terén ezek a legfontosabb felhasználások. A hangfelismerés elsődleges mérőszáma a felismerés pontossága. Ez a hangüzenet pontos megértését jellemzi. Ha a beszélőnek egyedi kiejtése vagy erős akcentusa van, akkor egyszerűbb megérteni a beszédét, ha a rendszer ismeri azt, vagyis már hozzászokott annak beszéd stílusához. Vagyis a rendszer egyfajta szótárral, vagy hangminta készlettel rendelkezik az illetőről. Ez a megállapítás különösen igaz a számítógép alapú hangfelismerésnél, mely révén a hangfelismerés pontossága fokozható. [21]. Ennek megvannak a korlátai akkor, ha nem tudható előre, hogy ki fog beszélni az adott időpontban – pl. többen használnak egy gépjárművet. Bérautó esetén pedig ez az előny el is tűnik.

A szavakat, amelyeket a beszédfelismerő rendszernek le kell tudnia fordítani, szótárba kell foglalni [22]. Azoknál a beszédfelismerő programoknál, amelyek pl. számítógépes szövegbevitelre képesek, több ezer szavas – vagy ennél is nagyobb - szótár kell, hogy rendelkezésre álljon. A másik

véglet azon rendszerek, amelyeket arra terveztek, hogy a felhasználó például a feltett eldöntendő típusú kérdésekre pl igennel, nemmel vagy a felkínált opciók egyikével válaszoljon. Ezeket hívjuk kis szótáras rendszereknek. Az autóipar területén a felhasználás célja határozza meg, melyik megoldást kell használni. A modern rendszerekben, kihasználva a mobil távközlési rendszerek és a felhő alapú számitástechnikai rendszereket, a szótár mérete – és a nyelvek száma – már nem jelent korlátot.

A gépi beszédfelismerés az alábbi fázisokra bontható:

- Felvétel

  A hang vagy beszéd jel eljuttatása az azonosító rendszernek.
- Előzetes szűrés / lényegkiemelés

  A beszédjelből megkísérli meghatározni a beszéd tartalmát, és kiküszöbölni a felismerés szempontjából érdektelen információkat (zaj, fázis, torzítások). A kimenet – matematikai módon leírva - gyakorlatilag egy adott dimenziójú lényegvektor-sorozat.
- Keretezés

  Ebben a fázisban megpróbáljuk az elemi egységekre, fonémákra bontást, pl 10-30 ms hosszakban, 50%-nál nem nagyobb fedésben.
- További szűrés

  Nem minden esetben szükséges, csak akkor, ha az előzetes szűrés nem ad megfelelő eredményt. Sokszor ez a lépés végzi pl az időillesztést vagy hangerőre normalizálást.
- Összehasonlítás és osztályozás

  Ezen fázis célja felismerni a jelentést. Ez legtöbb esetben mintával való összehasonlítást jelent (például két jellegvektor összehasonlítása). Minden itt alkalmazott módszer egy valószínűségi és pontossági illesztést végez.
- Kimenet kezelése

  Ez a lépés a felhasználási céltól függ, sokszor nem is érzékelhető külön, de ez teszi lehetővé további DL/ML módszerek alkalmazását, mivel numerikus értékek rendelhetők itt a feldolgozott szöveg elemeihez.

Az összehasonlítás és illesztés lépés úgy valósítható meg a leghatékonyabban, hogy a mintát, azaz a bejövő vektorsorozatot - statisztikai úton becsült - valószínűségi modell struktúrákhoz illesztjük, és a legjobb illeszkedés javára döntünk - Viterbi-approximációval. A legjobb illesztés keresését szokták általánosságban *keresésnek* vagy *dekódolásnak* nevezni, és az ahhoz tartozó szósorozatot pedig a beszédfelismerés eredményének.

Formálisan:

$$\widetilde{W} = arg_w \max P(W|O) \tag{4.1}$$

ahol $W = w_1, \ldots, w_K, K \in N$ egy megengedett (modellezett) szósorozatot jelöl, és $O = o_1, \ldots, o_T$ a bejövő beszédjel T elemű lényegkiemelt vektorsorozatát jelöli. $\widetilde{W} = \widetilde{w}_1, \ldots, \widetilde{w}_{\widetilde{K}}$, $\widetilde{K} \in N$ pedig a felismert szósorozatot jelenti.

(1) a Bayes-szabály segítségével a következőképpen alakítható át:

$$\widetilde{W} = arg_w \max P(W) * P(W|O) \tag{4.2}$$

A (2) egyenletet a beszédfelismerés MAP alapegyenletének is szokták nevezni. A formula szemléletesen választja szét az adott szósorozatnak a nyelv által önmagában becsült valószínűségét, $P(W)$-t, az akusztikai megfigyelés valószínűségétől, $P(O|W)$-től. Az előbbi valószínűséget a *nyelvi modell* adja, az utóbbit az *akusztikus modell*. A beszédfelismerésnél a mintaillesztés feladata tehát nem más, mint adott nyelvi és akusztikus modell, valamint bejövő akusztikus jellemzővektor-sorozat esetén a legnagyobb valószínűségű szósorozat megtalálása [23]. A nyelvi modellel – amely leginkább alapoz a gépi tanulás módszereire -, az 5. fejezetben foglalkozom részletesen.

4.2.1. Az akusztikus modell

Az akusztikus modellt, tradicionálisan fonológiai (szó-->fonéma) kiejtési, és fonéma szintű akusztikus modellekre szokták bontani. Ezek részletes matematikai leírása megtalálható [23]-ben, most csak a definiciókat emelem ki.

„Fonológiai kiejtési modell*: A fonológai kiejtési modell az egyes szósorozatokhoz valószínűségekkel ellátott fonémasorozatokat rendel. Ez tipikusan egy kiejtési szótár segítségével történik, melyben az egyes ortografikus szóalakokhoz több, valószínűséggel ellátott fonéma sorozat is tartozhat.* [23, p. 12]"

A minta illesztése az ML (Maximum Likelihood) értelemben optimális szó-fonéma részsorozat összerendelést is magában foglalja. A kiejtési minták összessége, a szótár előállítása történhet kézi módszerrel – pl. a mintaszavak egyenkénti rögzítésével -, vagy automatikus módszerekkel, pl. nagymennyiségű szó vagy mondat tanításával.

„Fonéma akusztikus modell*: A fonéma szintű akusztikus modell feladata adott (megengedett) fonémasorozathoz valószínűséget rendelni az akusztikai megfigyeléssorozat alapján.* [23, p. 12]"

Ez utóbbi modellt tovább szükséges osztani a környezet hatásának lekezelése miatt. A *környezetfüggetlen fonéma akusztikus modellek* esetén a fonéma kiejtési modell a szó kiejtési modellhez hasonlóan környezettő függetlenül szótár formájában is megadható és használható a felismeréshez. Itt a leképezés egyértelmű, azaz egy fonémához egy és csakis egy elemi akusztikus modell sorozat tartozik.

A *környezetfüggő fonéma akusztikus modellek* azon alapulnak, hogy az egyes beszédhangok artikulációja jelentős mértékben függhet a környező beszédhangoktól, bevett gyakorlat ezt explicit módon modellezni. Általánosságban egy adott kontextusban lévő fonéma környezetfüggő beszédhangmodelljét, a hangon felül annak c számú, bal és/vagy jobboldali szomszédja együtt határozza meg. A gyakorlatban c=1 használata terjedt el, mindkét szomszédos hang figyelembevételénél ezeket a hangmodelleket trifónnak nevezzük. Ha csak az egyik oldali szomszédot tekintjük, akkor a megfelelő oldali bifónnak, míg a környezetfüggetlen esetet, melyről az előző fejezetben beszéltünk monofónnak nevezzük.

4.3. A beszélő felismerése – hangfelismerés

Általános modellek és speciális felismerési körülmények modellek esetén adaptációnak nevezzük a modellparamétereknek a körülményekhez történő transzformálását nevezzük. Ennek egyik tipikus esete a beszélőadaptáció, amikor rendelkezésre áll egy nagyszámú beszélővel tanított beszélőfüggetlen akusztikai modell, de adott esetben tudható, hogy csak egyetlen – vagy kis számú - beszélő hanganyagán kívánunk beszédfelismerést végezni. A beszélőhöz adaptált akusztikus modellekkel a felismerés nem csak lényegesen pontosabb, de egyszersmind gyorsabb is lehet.
Az adaptációnak két alaptípusa használatos:

A *felügyelt adaptáció* során off-line módon történik a modellparaméterek átalakítása. Ilyenkor szükség van már rögzített és ismert adaptációs tanítóadatra (hanganyag + szövegátirat), mely alapján elvégezzük a transzformációt a modelleknek az adaptációs felvételekre történő jobb illeszkedését célozva.

A *felügyelet nélküli adaptáció* során nem ismeretes az adaptációs hanganyag tartalma. Ez a módszer ezért mindig két fő lépésből áll:

- felismerés adaptálatlan modellekkel, és így egy közelítő pontosságú szövegátirat előállítása
- ML lineáris regressziós modelltranszformáció elvégzése az előző pontban előállított átirat segítségével

Az adaptáció nyelvi, kiejtési, elemi akusztikai szintű is lehet. A fenti módszereknek komoly kihívásokat is meg kell oldani ami a beszéd természetességéből származik. A beszéd paraméterek számos hatás következtében megváltoznak. Változatosságuknak számos forrása van, amelyek a felismerési pontosságot jelentősen megnehezítik. Ezek a források lehetnek pl. a hangképző szervek eltérő biológiai jellemzői. Személyenként eltérő méretek, de minden olyan eltérés, ami a fizikai produktum jellemzőit is megváltoztatja [22, pp. 2-5].

A hangképzés folyamatosan változó mozgások összessége, amelyet a felismerésnél egységekre bontva használunk. A folyamatos hangképzőszervi mozgások miatt egyik hang fizikai jellemzői befolyásolják az azt megelőző és követő hangok fizikai jellemzőit. Ezt nevezik koartikulációs hatásnak. A magyar nyelvben is jellemző ez, de a dupla mássalhangzók is jó példák erre.

A környezeti, akusztikai körülmények is okozhatják a fizikai paraméterek variáltságát. Ilyenek pl. a zajos, zajtalan környezet, visszhangok, termek, telefonbeszéd stb. A variáltság csoportosításakor elsősorban a beszélőkön belüli, és a beszélők közötti változatosság illik a legjobban a felismerők működési tulajdonságaihoz.

A beszélőn belüli variáltság fő okai a koarticuláció, a ritmus, hangmagasság, hanglejtés, hangerő, nyomaték béli különbségek. Betegség, megfázás, de akár a fizikai fittség aktuális állapota is igen nagymértékben megváltoztatja a hangok akusztikai paramétereit. Környezeti hatások, izgalom, meglepetés, a környezet zajossága, de a beszélők közötti szociális viszonyok is szintén erősen befolyásolják a beszéd akusztikai tulajdonságait.

A beszélők közötti variáltság fő oka a biológiai tényezők pl. a beszédképző szervek méretkülönbsége, ami az akusztikai paraméterek jelentős variáltságát okozza női, férfi vagy gyermekhangok esetében, de egy csoporton belül is. Környezeti hatások két csoportban: a statikus (teremakusztikai hatások, utózengési idő, rögzítő berendezések, stb.) és dinamikus (zaj, mikrofon pozíció stb.) hatások szintén erősen befolyásolják a beszéd akusztikai paramétereit.

Nyelvi különbözőségek, a tájszólás szintén forrásai a beszéd variáltságának. A fenti jellemzők mindegyike az anyanyelvre vonatkozik. Tanult nyelvek esetén ezeken felül több más tényező is befolyásolja a variáltság mértékét, ezáltal plusz komplexitást adva a felismeréshez.

5. A természetes beszédfelismerés módszerei

A gépi beszéd és beszélő felismerési eljárások lényegében két, egymástól jól elkülöníthető elméleti alapra épülnek. Az egyik a statisztikai alapú információelméleti megközelítés, a másik a szabálybázisú, kognitív módszer.

- Szabálybázis alapon működnek pl. a különböző szakértői rendszerek.
- Statisztikai alapú feldolgozást használnak a *Rejtett Markov Model* (**Hidden Markov Model: HMM**), vagy *neurális hálózatok* (**Neural Network, NN**) használatával megvalósított felismerő rendszerek.

Sok, a gyakorlatban megvalósuló sikeres beszélő és beszédfelismerő rendszer statisztikai alapokon működik [22]. Ezekhez is azonban szükségesek adatbázisok, mivel a természetes beszédfelismerés első lépése a hang konverziója a számítógépek számára használható, numerikus alapú információkká. Ehhez pedig adatbázisokra van szükségünk.

Az ilyen adatbázisoknak tartalmazni kell azokat a megfigyeléseket, amelyek a paraméterbecsléshez szükségesek, mindazokat a mintákat, amelyek egységesen lefedik a beszéd (és a környezeti zajok) variáltságát. Így ha egy beszédhang nincs benne a tanításhoz használt adatbázisban, akkor azt a beszédhangot sohasem vagy permanensen hibásan fogja a gép felismerni.

A gyakorlatban az adatbázisok tervezésénél – a méret, rendelkezésre álló számítástechnikai, tárolási kapacitás, a felhasználás során használható adatátviteli sávszélesség, stb mellett - figyelembe kell venni, hogy az adatbázis létrehozása nem más, mint a véletlenszerű folyamat egyes megvalósulásainak összegyűjtése.

Fontos meghatározni előre a felismerés elvárt pontosságát is, vagyis mennyire kritikus az esetleges hibás felismerés következménye. Egy szimpla beszédet szöveggé konvertáló alkalmazásnál, átlagfelhasználó számára, megengedhető nagyobb hibaszázalék, mivel maga a felhasználó képes lehet a hibás felismerés korrigálására. Gépi vezérlő rendszereknél, a felismerés után használt szabályokkal csökkenthető a hibás felismerés következménye, pl. a folyamat adott pontjába nem illeszkedő parancs esetén a rendszer megerősítést kérhet. Viszont autóvezetés közben, a hibázás esélyét nullára kell szorítani, a lehetséges súlyos következmények miatt.

A beszéd felismeréséhez használt referencia adatbázisok, általában szoftver segítségével generált, digitálisan tárolt és a további használathoz elengedhetetlenül szükséges magyarázó jegyzetekkel, címkézésekkel ellátott beszédfelvételek gyűjteményei. Ma már jellemzően ezek – bizonyos kritikus rendszerek kivételével – felhőalapú rendszerekben kerülnek tárolásra, és maga a felismerés folyamata is a felhőben történik.

A rendelkezésre álló, helyben meglévő kapacitás függvénye a feladatok megosztása, és itt is fontos döntési tényező, hogy milyen szinten működjön autonóm módban a helyi rendszer. Ezt pl. a tanítás és a használat során gyakran felmerülő minták – pl. a jellemzően a kocsit használó sofőr beszédmintái – helyben történő tárolásával és azonosításával tudjuk megoldani.

A használt adatbázisokat 3 alapvető csoportra szokták osztani, ebből pl. az utolsó típus lehet kimondottan alkalmas pl. gépjárműben történő felhasználásra:

- *Diagnosztikus adatbázis*: a nyelvi, fonetikai kutatások segítésére szolgál.
- *Általános célú adatbázis*: általános szótárakat tartalmaz mintaként, sokfajta felhasználásra alkalmas.
- *Speciális adatbázis*: olyan beszédminták gyűjteménye, amely meghatározott felhasználási területen való alkalmazás készül.

A számunkra legfontosabb felhasználási terület ezen munkában a beszédfelismerés. Ennek a fejezetnek az alapjai a [24]-ből származnak, de több más munka is felhasználásra került. Az alábbiakban a felismerés nyelvi modelljeivel foglalkozunk, az akusztikus alapok korábban lettek áttekintve. Megállapítható, hogy a sikeres és hatékony beszédfelismerő rendszereknek kombinálni szükséges a minta alapú felismerést, a modern neurális hálózatokkal és mesterséges intelligencia nyújtotta megoldásokkal.

5.1 Statisztikai nyelvi modellek

A statisztikai nyelvi modellek, valószínűséget rendelnek a szavak sorrendjéhez. Nyelvtanilag helyes esetben a szavak sorrendje nagy valószínűséggel meghatározó, főként ha az azonos vagy hasonló sorozatok sokszor fordulnak elő. Az első lépés az ezekhez tartozó valószínűségek kiszámolása. Ez a valószínűség a legegyszerűbb módon az *5.1 képlettel* számítható ki, ahol a w_x a szavak sorozatát határozza meg:

$$\mathrm{P}(w_1,\ldots w_m)=\prod_{i=1}^{m}\mathrm{P}(w_i|w_1,\ldots,w_{i-1}) \tag{5.1}$$

Sajnos az értéket nehéz becsülni főként, ha az *i* értéke kellően magas. A Bayes szabály alapján átalakítva, az egyik megoldás az egyszerűbb kiszámításra a *Markov-sejtés* használata, amelyet a gépi tanulás területén *rövid memória* sejtésnek is hívnak, amellyel az érték így is meghatározható (az n a zsetonok száma):

$$P(w_i|w_1,\ldots,w_{i-1}) \approx P(w_i|w_{i-n+1},\ldots,w_{i-1}) \tag{5.2}$$

A képlet jobb oldala a következő módon számítható ki, ami nagy *n* érték esetén elméletileg jobb különbséget ad, de a nagy adatmennyiség nem feltétlenül teszi lehetővé a megfelelő becslést:

$$P(w_i|w_1,\ldots,w_{i-1}) \approx P(w_i|w_{i-n+1},\ldots,w_{i-1}) = \frac{Count(w_{i-n+1}\ldots w_i)}{Count(w_{i-n+1},\ldots,w_{i-1})} \tag{5.3}$$

Ha bigram modellről beszélünk, vagyis n=2, a képletünk igy néz ki:

$$P(w_i|w_1,\dots,w_{i-1}) \approx P(w_i|w_{i-1}) = \frac{Count(w_{i-1,}w_i)}{Count(w_{i-1})} \tag{5.4}$$

Az n-gram esetben, finomítást, tisztítást (smoothing) kell alkalmaznunk a használhatóság érdekében. Ezzel a megoldással az (5)-ös képlet a következő módon egyszerűsíthető, ahol β értéke határozza meg a regularizáció/simítás mértékét és d a szótár mérete :

$$P(w_i|w_{i-1}) = \frac{Count(w_{i-1,}w_i)+\beta}{Count(w_{i-1})+d*\beta} \tag{5.5}$$

Ennyi alapozás után térjünk át a specifikus modellekre, feladatokra.

5.1.1. A Continuous Bag-of Word modell

A Continuous Bag-of-Word (CBOW) modellel azt a folyamatot próbáljuk reprezentálni, amikor egy konkrét szóra következtetünk az környezetéből [25]. Másképpen, szeretnénk helyesen kiegészíteni a hiányzó szóval egy adott szövegkörnyezetet. Ezt egy olyan – egy bemeneti, egy rejtett és egy kimeneti réteget – tartalmazó neuron reprezentálja, amelyből 1 bemeneti csomópont hiányzik. Szükséges megadni, hogy hány szót használunk a hiány előtt és után, a célfüggvény kiszámolásához.

Ebben a modellben szoftmax függvény segítségével határozzuk meg a $p\left(w_n\middle|w_{n-c}, w_{n-(c-1)}, \dots w_{n-1}, w_{n+1}, w_{n+2}, \dots w_{n+c}\right)$ feltételes valószínűségeket. Azt, hogy mekkora a valószínűsége annak, hogy az w_n-t határoló szavakból helyesen következtetünk rá.

A módszer legegyszerűbb verziója a bigram módszer – lásd korábban – ahol csak a szomszédos szavakat használjuk.

5.1.2. A Skip-gram modell

Ezzel a modellel pl. olyan típusú problémákat próbálunk megoldani, amikor szeretnénk egy felismert szóból annak környezetére következtetni. Ennek sok előnye lehet a beszédfelismerés során, a sebesség lehetséges növelésén túl, pl. amennyiben a következtetés nem helyes – pl. a következő szó kiesik az elvárt tartományból – ez utalhat arra, hogy nem teljes felvétel lett analizálva, valami kimaradt.

Az alap Skip-gram modell egy bemeneti, egy rejtett és egy kimeneti réteget tartalmaz, ahol az első kettő csak 1-1 csomóponttal rendelkezik. Az alap Skip-gram modellben szoftmax függvénnyel határozzuk meg a $p(w_{n+t} \mid w_n)$ feltételes valószínűségeket. Ez azt adja meg, hogy mekkora a valószínűsége annak, hogy az n-edik szóból az n + t-edik szóra következtetünk.

A modell alapmegvalósításának nagy problémája, hogy ilyen módon megvalósítva nagyon költséges a modell, mivel összetettsége a szótár méretének többszöröse. A szótárak jellemzően hatalmas méretűek, így szükséges pár olyan módszer, amellyel hatékonyabbá tehető.

5.1.3. A GloVe modell és kiterjesztése

A globális vektor (GloVe) modell célja, hogy ötvözze a nyelvfeldolgozáshoz használt eszközök két nagy csoportját, a globális mátrix faktorizációt és a lokális szövegkörnyezetet használó eljárásokat. A modellt egy 2014-es publikáció alapján vizsgálom meg [26].

A modell bevezetésekor említettem, hogy a modell célja, hogy a globális és a lokális információkat használó eljárásokat egybevonja, ezzel javítva ezeket. A korábban láthattuk, hogy a Skip-gram modell is egy lokális szövegkörnyezetet használó eljárás. Emellett a korpusz statisztikai adatait használja, hogy meghatározza a szavak vektor reprezentációját és a tanítás felügyelet nélkül zajlik.

Az optimalizálni kívánt célfüggvény megadása helyett, az alábbiakban levezetem, hogy milyen módon lett megalkotva. A faktorizáció során használt mátrixot C-szel fogjuk jelölni, azaz az C_{ij} elem jelentése, hogy a *j* szó hányszor fordul elő az *i* szó környezetében. $C_i = \sum_k C_{ik}$-vel fejezzük ki, hogy hány másik szó fordul elő az *i* szó környezetében, míg $P_{ij} = P(j \mid i) = C_{ij}/C_i$ megadja *j* szó megjelenésének valószínűségét az *i* környezetében. A szavak jelentését bizonyos módon kifejezik a definiált együttes előfordulásra vonatkozó valószínűségek [8]. Hogy ki tudjuk használni ezt a tulajdonságot, ezért bevezetek egy F függvényt a hányados kiszámításához, melyben $w \in \mathbf{R}^N$ szóvektorok és $\tilde{w} \in \mathbf{R}^N$ különböző szövegkörnyezet béli szóvektorok, N pedig a szótár méretét jelöli.

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{5.6}$$

A jobb oldali értéket megkaphatjuk a korpuszból, míg a baloldali kifejezést többféleképpen is definiálhatjuk. Szeretnénk, ha F a szavak vektorterében reprezentálná az információkat, amiket a hányados tartalmaz. Mivel a vektorterek lineáris struktúrák és az argumentumok is a kívánt vektortér elemei, tekinthetjük a két vizsgált szóvektorunk különbségét is a következő módon:

$$F\big((w_i - w_j), \widetilde{w}_k\big) = \frac{P_{ik}}{P_{jk}} \tag{5.7}$$

A linearitás megtartásához a kiértékelés során – mivel a keresett függvény nagyon összetett is lehet -, a (5.7)-et átalakítjuk. Egyenletben szereplő argumentumok skaláris szorzatát kell használnunk [8].

$$F\big((w_i - w_j)^{\,T} \widetilde{w}_k\big) = \frac{P_{ik}}{P_{jk}} \tag{5.8}$$

A további átalakításokkal megkapott modellnek a hátránya, hogy egyenlően súlyozza az együttes előfordulásait a szavaknak, attól függetlenül, hogy ezek milyen gyakorisággal történnek meg. Ennek eredménye, hogy a ritka adatok zajosak és kevesebb információt tartalmaznak, mint a gyakoriak. Emellett az C mátrix nagyon ritka lesz. Ezen eset elkerülésének egyik lehetősége a legkisebb négyzetek módszerét használó regressziós modell:

$$J = \sum_{i,j=1}^{N} f\big(C_{ij}\big)\,(w_i^T\,\widetilde{w}_k + b_i + \tilde{b}_k - \log C_{ik})^{\,2} \tag{5.9}$$

ahol f egy olyan súlyfüggvény, amelyre a következő feltételek teljesülnek:

- f (0) = 0. Ha f folytonos függvény, akkor x elég gyorsan tart 0-hoz, hogy $\lim_{x\to 0} f(x)\ log^2 x$ véges legyen.
- f (x)-nek nemcsökkenőnek kell lennie ahhoz, hogy elkerüljük a ritka együttes előfordulások túlsúlyozását.
- f (x)-nek kicsinek kell lennie a nagy x értékekre, hogy a gyakori együttes előfordulások ne legyenek túlsúlyozva.

A fenti számolás módok megfelelő programozási háttérrel használhatók.

5.2. Az adatbázisok és szerepük

A beszédfelismerés fontos eleme az adatbázis, amit a tanításhoz, majd a működéshez használunk. Szöveg felismerése esetén ezeket szótárnak nevezzük, és méretük a feladat bonyolultságától függ. Egy kizárólag szaknyelvet – pl. orvosi vagy jogi – fordító rendszerhez elég azoknak a kifejezéseknek a szótára, amelyek itt felmerülnek. Amennyiben egy kérdés-felelet típusú felhasználáshoz készítünk adatbázist, meghatározottak a kérdések, és előre definiálható a várható válaszok készlete is. Viszont ha egy általános célú beszédfelismerőt szeretnénk készíteni, ahhoz nagyméretű szótár kell. A legtöbb rendszerben többféle szótár együttes használatáról van szó, ahol pl. külön adatbázis van a fonémák azonosítására – ami adott esetben magának a nyelvnek a felismerését is szolgálja - és egy másik a szavak, mondatok vagyis a tartalom számára [8].

Az autóipari felhasználások többsége is behatárolható szótárral rendelkezik. A felhasználás célja alapján – pl. egy térkép alkalmazás kezelésénél – ez a szótár már nagyobb, de pl. a kényelmi funkciók hangvezérléséhez már nem szükséges nagy méret.

5.3. Szűrési módszerek

A beszédfelismerés egyik fontos a szűrés, amikor a tartalmat szeretnénk meghatározni, nem az egyes szavak minta szerinti felismerésével – amihez nagy szótári adatbázis kellene -, hanem a szavak kapcsolatát kihasználva valószínűsíteni a következő szó jelentését. Erre többféle módszer létezik – nincs is mód az összes ismertetésére -, így most néhányat ismertetek röviden, a főbb előnyök és gondok ismertetésével együtt [8].

Amikor a szűrés során vizsgálódunk, a beágyazódás vizsgálatához 3 alapmódszer használható:

- *Kernel-alapú beágyazódás*: Ilyenkor mondat vagy teljes szöveg alapján próbáljuk kalkulálni az adott szó beágyazódását. Beszélt szöveg felismerésénél maximum a mondat, vagy a mintaként szolgáló felvétel lehet az alap, nincs mód a teljes szöveg kezelésére.
- *Szétosztott jelentéstani modell*: Ezek általános a számolós módszerek, ahol a szó előfordulásának száma alapján következtetünk a jelentésére a felismerésnél. Amikor hangmintákat elemzünk, a módszer segíthet a gyorsabb találatban azzal, hogy az adott helyzetben legvalószínűbb mintára mutat [8].
- *Összefüggéses neurális-hálózat modell:* A modell hasonló az előzőhöz, sok tekintetben, A neurális hálózat használatával itt a szavak és környezetük leírását egy felügyelt tanulási feladatra alakítjuk. A különböző módszerek között sok a hasonlóság [27].

- *Visszacsatolt neurális-hálózat modell:* Ez a módszer talán a legfejlettebb ezen a téren, ami akár mondat-beágyazódás szinten képes analizálni. Beszédanalízis során segíthet a mondat pontos értelmezésében, ezért is használják pl. a valós idejű műsorfeliratozáshoz.

5.3.1. A szó-jelentés mátrix faktorizációs modell

Az egyik nagy csoport, amikor szó-kontextus mátrixot számol ki a modell, az egyes célszavak kontextuális elemeinek száma alapján. Ehhez a cél szó előttről és utánról egyenlő számú szót használunk. Ez az ablak méret a tanító adatbázis méretétől függ, de általában nem nagyobb mint 10 (bigram … n-gram).

5.3.2. Mátrix felbontás számlálással – HAL modell

A legegyszerűbb módszer, ha az együtt előfordulási gyakoriságok mátrixát faktorizáljuk, hogy szavak beágyazódásait megállapíthassuk. A HAL modell (**Hyperspace Analogue to Language**) [28], egy régen létező, de csak az utóbbi években továbbfejlesztett és optimalizált módszer erre.

Minden *i* szóra bevezetjük a c_{ij}-t, ami a száma annak, ahányszor *j* együtt szerepel *i*-vel az ablakon belül. Az így kapott d x d – ahol d a szótár mérete – első lépésben bontásra kerül p-rendű U és V mátrixra:

$$C \approx U V^T \tag{5.10}$$

Mivel sok zéró szerepel az értékek között – hiszen sok szó nem szerepel a másik környezetében egyáltalán, így a legjobb választás a sztochasztikus gradiens módszer, ami csökkenti a 0-ák szerepét. Definiáljuk a következő közelítést:

$$c_{ij} \approx \hat{c}_{ij} = \bar{u}_i * \bar{v}_j \tag{5.11}$$

Itt használjuk a torzított faktorizációt (bias factorization), b_i^r értéket a sorokon, b_j^c értéket az oszlopokon.

$$\hat{c}_{ij} = b_i^r + b_j^c + \bar{u}_i * \bar{v}_j \tag{5.12}$$

A mátrix faktorizáció a következő célfüggvényt adja:

$$Minimize\ J = \sum_i \sum_j \left(c_{ij} - b_i^r - b_j^c - \bar{u}_i \cdot \bar{v}_j \right)^2 + \lambda \cdot \left(\sum_i \left|\left|\bar{u}_i\right|\right|^2 + \sum_i (b_i^r)^2 + \sum_i \left|\left|\bar{v}_j\right|\right|^2 + \sum_i (b_j^c)^2 \right) \quad (5.13)$$

Az U és V mátrixok véletlenszerű inicializálásával szükséges futtatni a modellt a tanítás során, amig a célfüggvény el nem éri a kitűzött értéket. Különböző utófeldolgozási módszerek használhatók az eredményekre (pl. neurális hálózatok). Általánosságban megállapítható, hogy a módszerek kombinációja – egymás után használva azokat – adja a legjobb eredményt.

5.3.3. PPMI Mátrix felbontás

A korábban ismertetett GloVe modell logaritmikus normalizációt használ, a nagyon gyakran használt szavak hatásának csillapításához. A másik megoldási lehetőség erre a kapcsolatok vizsgálata a mátrix előállításához. Ilyen például a *positive pointwise mutual information (PPMI)* modell [29]. Ez a módszer hasonló módon ábrázolja a szavak környezetét és kapcsolatait és az *5.14 egyenlettel* írható le:

$$PPMI(i,j) = max\left\{ \log\left(\frac{N \cdot c_{ij}}{f_i \cdot q_j} \right)_{\uparrow} 0 \right\} \quad (5.14)$$

ahol $f_i = \sum_j c_{ij}$ és $q_j = \sum_i c_{ij}$.

5.3.4. A módosított PPMI Mátrix felbontás

Amennyiben egy k értékkel módosítjuk az egyenletet, a módosított PPMI (SPPMI) modellt kapjuk, aminek speciális esete a k=1, ami az alap modellt adja.

$$SPPMI(i,j) = max\left\{ \log\left(\frac{N \cdot c_{ij}}{f_i \cdot q_j} \right) - \log(k)_{\uparrow} 0 \right\} \quad (5.15)$$

k>1 értékeket használva jobb eredményeket kapunk. Ilyenkor az egyébként népszerű – és később bemutatásra kerülő – negatív mintavételezéses word2vec módszerhez hasonló eredményeket kaphatunk.

Tovább finomíthatók az eredmények, ha szintaktikai kapcsolatok is számolunk a mátrix felállításánál. Ebben az esetben a mátrix oszlopa nem kell hogy megegyezzen a szóval, elég ha bármilyen kapcsolat fennáll az adott szóval. Hozzárendelhetünk az egyes szavakhoz nyelvtani jellemzőt (pl *ír-I* vagy *só-F*), ahol azt jelezzük, hogy igeként vagy főnévként szerepel az adott szó.

Ehhez mindenképpen szükséges NLP alapú előfeldolgozás, hiszen pl. az *ír* szó jelentheti az írás igéjét, de jelenthet egy Írországból származó – vagy ilyen felmenőkkel rendelkező – személyt is, és ez külön cimkét feltételez.

5.4. Neurális kiterjesztés *word2vec* módszerrel

A *word2vec* egy lokális szövegkörnyezetet használó vektortér modell. Ennek 2 fő variánsa használatos – mindkettő szerepelt már [24, p. 331]:

- *A célszó meghatározása a szövegkörnyezetből*: Ez a modell megpróbálja meghatározni az *i*-dik szót, w_i-t az előre meghatározott méretű szövegkörnyezetből. Vagyis a $w_{i-t}w_{i-t+1} \dots w_{i-1}w_{i+1} \dots w_{i+t-1}w_{i+t}$ szókészletet használjuk a w_i szó meghatározásához. Ez a *continuous bag-of-words* (CBOW) módszer.
- *A szövegkörnyezet meghatározása a célszóból*: Ez a skip-gram modell, ahol a tartalomra próbálink következtetni az *i*-dik szó meghatározott szélességű környezetéből. Ezt a következtetést, 2 módon végezhetjük el. A multinomial technika 1 szót határoz meg a *d* elemű szótárból. A másik a Bernoulli módszer, amely a negatív mintavételezést használja a hatékonyság növelésére.

A továbbiakban azt mutatom be, milyen matematikai módszerekkel lehet ezen modellek hatékonyságát javítani, csökkentve a számítási igényt is..

5.4.1. A skip-gram javítása neurális hálózattal

Az alábbiakban ismertetem a főbb lépéseit annak, hogyan javítható a skip-gram modell, neurális hálózat felhasználásával. Ez a modell is egy bemeneti, egy rejtett és egy kimeneti réteget tartalmaz. Ezek a rétegek mátrixként vannak ábrázolva, és amennyiben *d* elemű a szótárunk, a bemeneti réteg egy *d x p* méretű mátrix, míg a kimeneti réteg *m* darab *p x d* méretű mátrixok csoportja, ahol *p* a rejtett réteg mérete [24, pp. 334-336].

A rejtett réteg kimenetét a d x p mátrix $U[u_{jq}]$ súlyozásának felhasználásával írhatjuk le:

$$h_q = \sum_{j=1}^{d} u_{jq} x_j \quad \forall q \in \{1 \dots p\} \tag{5.16}$$

A $\hat{y}_{ij}$ kimenet a valószínűsége annak, hogy az *i*-dik pozíciójú szó, a *j*-edik szót adja a lexikonban. Mivel a kimeneti réteg minden csoportja ugyanazt a *p x d* méretű mátrixot használja, a neurális hálózat ugyanazt a multinomiális eloszlást feltételezi minden szóra, igy a következőket kapjuk:

$$\hat{y}_{ij} = P(y_{ij} = 1|w) = \frac{exp\left(\sum_{q=1}^{p} h_q v_{qj}\right)}{\sum_{k=1}^{d} exp\left(\sum_{q=1}^{p} h_q v_{qk}\right)} \; \forall i \in \{1 \dots m\} \tag{5.17}$$

Itt kell megjegyezni, hogy $\hat{y}_{ij}$ valószínűség nem változik, ha változtatjuk *i*-t, fix *j* mellett, mivel a fenti egyenlet jobb oldala nem függ az *i*-dik szó pontos helyétől a vizsgált ablakban. A visszaterjesztési algoritmus veszteségfüggvénye a negáltja a valószínűségek logaritmusának, a következőképpen felírva:

$$L = -\sum_{i=1}^{m} \sum_{j=1}^{d} y_{ij} \log\left(\hat{y}_{ij}\right) \tag{5.18}$$

A fenti egyenlőséget javíthatjuk a tanulási rátával, ezzel egyszerűsítve a számítási feladatokat:

$$\bar{u}_i \Leftarrow \bar{u}_i - \alpha \frac{\partial L}{\partial \bar{u}_i} \; \forall i \tag{5.19}$$

$$\bar{v}_j \Leftarrow \bar{v}_j - \alpha \frac{\partial L}{\partial \bar{v}_j} \; \forall j \tag{5.20}$$

A számítás elvégezhető deriválással, de anélkül is. Ez utóbbi módszerrel folytatom.

A hibázás valószínűsége a meghatározásnál a szótár j-dik szavának az i-dikkel összefüggésben, legyen $\left|y_{ij} - \hat{y}_{ij}\right|$. Viszont előjeles hibát, ϵ_{ij} használunk, amely csak akkor javasolja a szót, ha értéke pozitív. Ezzel elhagyhatjuk a modulust és igy a következő marad:

$$\epsilon_{ij} = y_{ij} - \hat{y}_{ij} \tag{5.21}$$

Ezután a konkrét *r* bemeneti szó és a kimeneti tartalom meghatározása így változik:

$$\bar{u}_r \Leftarrow \bar{u}_r + \alpha \sum_{j=1}^{d} \left[\sum_{i=1}^{m} \epsilon_{ij}\right] \bar{v}_j \qquad \text{[csak az r bemeneti szóra]}$$

$$\bar{v}_j \Leftarrow \bar{v}_j + \alpha \bar{v}_j \left[\sum_{i=1}^{m} \epsilon_{ij}\right] \bar{h} \qquad \text{[minden szóra a szótárban]}$$

Itt az $\alpha > 0$ a tanulási ráta. Az U mátrix p-dimenziójú sorai a szó kiterjesztéseként értelmezettek. Másképpen a konvenció az, hogy a bemenet kiterjesztése használt az U mátrix sorain, a kimenet kiterjesztése helyett a V mátrix oszlopain. Azt, hogy melyik módszert célszerű használni, a feladat dönti el [30], mindkettő jár előnyökkel és hátrányokkal. Gyakran a két irány kombinálása vezet eredményre.

5.4.2. A skip-gram és CBOW modellek kiterjesztése, optimalizálása

A hierarchikus szoftmax modell megoldás arra, hogy hogyan optimalizáljuk a kimeneti vektorok (v'_j) javításának a számítási igényét a Skip-gram vagy a Continuous Bag-of-Word modellekben [31]. Itt a modell bináris fában kódolja az összes szót a szótárból. Ez az N szó a gráf leveleiben helyezkedik el, ezeken kívül még N - 1 belső csúcs van (a gyökérrel együtt) [8].

A belső csúcsokra a következő jelölést használjuk: *n(w, j)*, ahol a *j*-edik csúcsot jelöli a gyökérből a *w* szóhoz vezető elérési úton. Mindig a gyökér számít az első csúcsnak és minden csúcsba egyértelmű az út a gyökérből. Ezeket az utakat tudjuk arra használni, hogy megbecsüljük a valószínűségét a levelekben lévő szavaknak. A hierarchikus szoftmax modell egyik eltérése a korábbiakhoz képest, hogy itt a belső csúcsokhoz definiálunk kimeneti vektorokat: $v_{n(w,j)}$ az *n(w,j)* belső csúcshoz tartozó kimeneti vektor. Annak a valószínűségét, hogy a *w* szó az elvárt kimenet, így definiáljuk [8, p. 32]:

$$p(w|w_0) = \prod_{j=1}^{L(w)-1} \sigma\left(b\left(n(w,j+1) = ch\big(n(w,j)\big)\right) \cdot v'^{T}_{n(w,j)} \cdot h\right) \tag{5.22}$$

ahol *L(w)* a gyökértől a *w*-be vezető út hosszát jelöli, *ch(n)* az *n* csúcs baloldali kapcsolatát, *h* a rejtett réteg kimeneti értéke és *b(x)* értéke 1, ha *x* igaz, különben -1. σ a szoftmax függvény, azaz *s(x) = 1/1 + exp(-x)* [8].

Tekintsük az ábrát és tegyük fel, hogy annak a valószínűségét keressük, hogy a w_2 a megjósolandó szó egy adott bemenet mellett. Erre a valószínűségre úgy tekintünk, hogy egy véletlen séta során mekkora a valószínűsége annak, hogy a gyökérből a w_2 levélbe jutunk. Ahhoz, hogy ezt meghatározzuk, szükségünk van arra, hogy milyen valószínűséggel lépünk egy belső csúcsból jobbra vagy balra. Ennek valószínűségét a következő módon definiálhatjuk:

$$p(n, left) = \sigma(v'^{T}_{n} \cdot h) \tag{5.23}$$

Ekkor annak a jobbra lépés valószínűsége:

$$p(n, right) = 1 - \sigma\left(v'^{T}_{n} \cdot h\right) = \sigma\left(-v'^{T}_{n} \cdot h\right) \tag{5.24}$$

Ahhoz, hogy meghatározzuk a belső csúcsok reprezentációját, vegyük a legegyszerűbb módozatát a modellünknek, amikor egy megadott szóból szeretnénk következtetni a következőre. Innen, több lépésben a korábbi levezetést használva megkaphatjuk az átsúlyozáshoz szükséges egyenleteket, erre terjedelmi okokból nem térek most ki [8].

5.4.3. Pontosság és hatékonyság

A fenti módszerek használhatósága több praktikus kérdést felvet. A beágyazott dimenziója a rejtett rétegnek segít középutat találni az kiugró elemek elnyomása és a változatosság között. A megnövelt dimenzió növeli a megkülönböztetést, de nagyobb adatmennyiséget is igényel. Általában százas nagyságrend használt a rejtett rétegben, de lehetőség van akár ezres méret használatára, amennyiben az alapszótárunk is nagyon nagy – vagyis a vizsgálandó minta.

A használt tartalom ablak általában 5 és 10 között mozog [24, p. 336], ahol tipikusan a skip-gram modell használja a nagyobb értékeket, míg a CBOW a kisebbet. Lehetőség van véletlen méretű ablak használatára is, ez azonban nagyobb súlyt ad azoknak a szavaknak, amelyek gyakran szerepelnek egymás mellett.

Másik gond a gyakori szavak – névelők, kötőszavak, létigék, stb – gyakori előfordulása a szövegben. Az egyik megoldás a *downsampling* – leminősítés - ezekre a szavakra, amikor csökkentjük a súlyukat, ami maga után vonja a mintavételi ablakok méretének növelését. Ezeket a gyakori szavakat sokszor kizárják a vizsgálatból.

5.4.4. Skip-Gram, negatív mintavételezéssel

Egy másik lehetőség a skip-gram és a CBOW modellek kiterjesztésére a negatív mintavétel használata, aminek a lényege, hogy nem az összes kimeneti vektort javítjuk, hanem csak egy részüket. Egy mintavételezést hajtunk végre rajtuk, és csak a kiválasztott példányokat frissítjük.

Az egyszerűsítés kedvéért most is tekintsük azt az esetet, amikor egyetlen bemeneti w_I szóból egyetlen szót szeretnénk megjósolni, ahol a w_O szót kapjuk eredményül. Jól működő modell esetén,

igaz lesz az, hogy w_I és w_O is a vizsgált ablakból ered, hiszen a CBOW és a Skip-gram modell esetén is a vizsgált szövegrész egy szavából következtetünk egy másik kifejezésére. Ebből kiindulva a negatív mintavételezésnél azt próbáljuk maximalizálni, hogy a w_I-w_O párok megtalálhatóak a vizsgált szókészletben. $p(D = 1 \mid w_I, w_O)$ jelöli azt, hogy a D szövegkörnyezet tartalmazza w_I és w_O szavakat, míg $p(D = 0 \mid w_I, w_O) = 1 - p(D = 1 \mid w_I, w_O)$ jelentése, hogy nem D-ből származik a bemeneti – kimeneti szópár. Ezek után a célfüggvény [8]:

$$\begin{aligned} \max \prod_{(w_I,w_O)\in D} p(D = 1|w_I, w_O) \\ = \max \log \prod_{(w_I,w_O)\in D} p(D = 1|w_I, w_O) \\ = \max \sum_{(w_I,w_O)\in D} \log p(D = 1|w_I, w_O) \end{aligned} \tag{5.25}$$

Használva a szoftmax függvényt és a szavak v_0 kimeneti vektor reprezentációját, az *5.26 célfüggvényt* kapjuk meg [8]:

$$\sum_{(w_I,w_O)\in D} \log \frac{1}{1+exp\left(-v'_{w_I}-v'_{w_O}\right)} \tag{5.26}$$

Ennek a függvénynek egy megoldása, ha $p(D = 0 \mid w_I, w_O) = 1$ minden szópárra úgy, hogy $w_O = K$, ahol K elég nagy szám. Viszont szeretnénk elkerülni ezt az esetet – a nagy számítási igény miatt is -, ezért változtatásra van szükségünk. Ha a célfüggvényben használunk olyan szópárokat is, amiknél $p(D = 0 \mid w_I, w_O)$ alacsony, kiküszöbölhetjük ezt a gondot. Ehhez elég, ha olyan szópárokat is számításba veszünk, amik nem a D korpuszból valóak. Erre az egyik módszer, ha véletlenszerűen generálunk egy szópárokból álló D_0 halmazt, melynél fenáll, hogy a szópárok nem részei D-nek. Az elnevezés is innen származik, hiszen negatív párosokból is válogatunk a mintavételezés során [8]:

$$\max \prod_{(w_I,w_O)\in D} p(D = 1|w_I, w_O) \prod_{(w\_I,w\_O)\in D'} p(D = 0|w_I, w_O) =$$

$$\max \prod_{(w_I,w_O)\in D} p(D = 1|w_I, w_O) \prod_{(w\_I,w\_O)\in D'} (1 - p(D = 1|w_I, w_O)) =$$

$$max \prod_{(w_I,w_O)\in D} \log p(D = 1|w_I, w_O) \prod_{(w\_I,w\_O)\in D'} \log(1 - p(D = 1|w_I, w_O)) =$$

$$max \prod_{(w_I,w_O)\in D} \log \frac{1}{1+exp\left(-v'_{w_I} v'_{w_O}\right)} \prod_{(w\_I,w\_O)\in D'} \log\left(1-\frac{1}{1+exp\left(-v'_{w_I} v'_{w_O}\right)}\right) =$$

$$max \prod_{(w_I,w_O)\in D} \log \frac{1}{1+exp\left(-v'_{w_I} v'_{w_O}\right)} \prod_{(w\_I,w\_O)\in D'} \log \frac{1}{1+exp\left(-v'_{w_I} v'_{w_O}\right)}$$

Ezen alkalmazva a $\sigma(x) = \frac{1}{1+\exp(-x)}$ jelölést megkapjuk [8]:

$$max \prod_{(w_I,w_O)\in D} \log \frac{1}{1+exp\left(-v'_{w_I} v'_{w_O}\right)} \prod_{(w\_I,w\_O)\in D'} \log \frac{1}{1+exp\left(-v'_{w_I} v'_{w_O}\right)} =$$

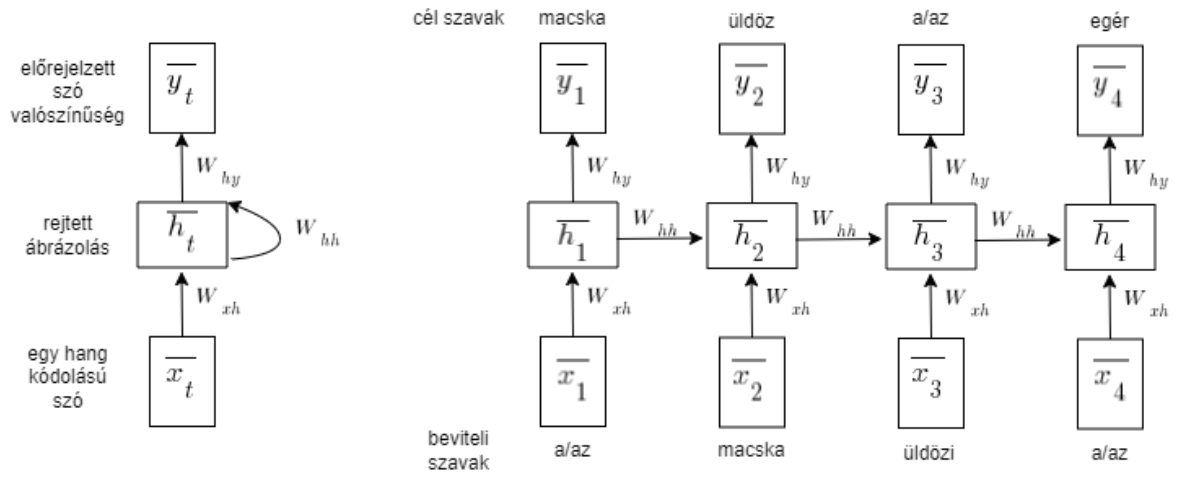
$$max \prod_{(w_I,w_O)\in D} \log \sigma\left(v'_{w_I} \cdot v'_{w_O}\right) \prod_{(w\_I,w\_O)\in D'} \log \sigma\left(v'_{w_I} \cdot v'_{w_O}\right) \quad (5.27)$$

5.5. Visszacsatolt neurális hálózatok a beszédfelismerésben

A visszacsatolt (recurrent) hálózat abban tér el a korábban már előrecsatolt társától, hogy van benne visszacsatoló hurok, azaz van olyan kapcsolat a hálóban, ami egy későbbi rétegből egy korábbiba vezet. Ezeket a visszacsatolásokat egyfajta memóriaként használja a hálózat a tanítási folyamat során. Ha több tanítási fázis van, akkor például a *f*-edik fázisban felhasználhatjuk a *f-x*-edikben kiszámított információinkat.

A visszacsatolt neurális hálózatokra jellemző, hogy a bemenet minták sorozata és a kimenet szintén megoldások sorozata. Az egyik lehetőség, hogy a bemenet szavak sorozata, míg a kimenet ugyanaz a sorozat 1 szóval eltolva. Szintén használt, hogy a bemenet szavak sorozata egy adott nyelven, míg a kimenet az ennek megfelelő szavak sorozata egy másik nyelven. A bemenet lehet komplett mondat is, ilyenkor a kimenet egy valószínűségi vektor, ami az utolsó szó feldolgozása után lesz teljes.

A *6. ábra*, ugyanannak a mondatnak az elemzését mutatja bal oldalon *visszacsatolt neurális hálózat* (RNN) és az *időrétegű* (time-layered) ábrázolási móddal a jobb oldalon [24, p. 344].



6. ábra. Mondat elemzés két módja.

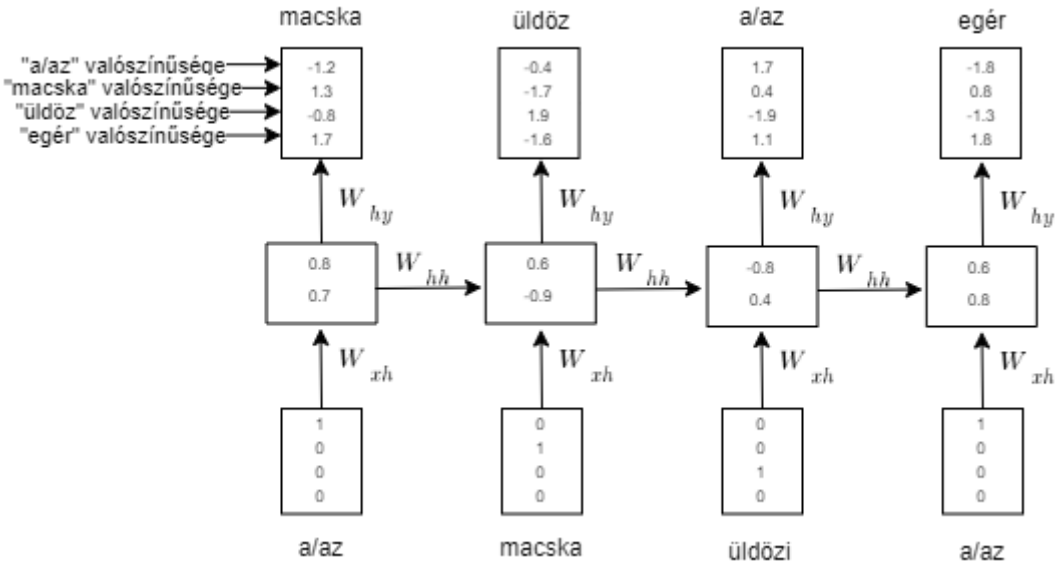
A jobb oldali reprezentáció közelebb áll a hagyományos hálózatokhoz, és megelőzi az esetleges végtelen ciklust, ami előfordulhat a bal oldali modellnél. A felső sorban a szó előfordulásának valószínűsége áll, az adott pozícióban. A második megoldásnál fontos észre venni, hogy a rejtett réteg mindig más műveletet rejt. Ennek a rejtett rétegnek matematikai reprezentációja alább:

$$\bar{h}_t = f\left(\bar{h}_{t-1}, \bar{x}_t\right) \tag{5.28}$$

Az ilyen típusú hálózatokat taníthatjuk egy speciális időalapú visszaszaporításos módszerrel (**Back Propagation Through Time**), amely a következő lépéseket tartalmazza:

- A bemeneti adatokkal sorban futtatjuk a modellt előre irányban és kiszámoljuk a hiba/veszteség arányt minden időpontra
- Kiszámítjuk a változást az éleket a súlyozásban, visszairányban annak elhanyagolásával, hogy a súlyozás az időrétegek között megosztott
- A megosztott súlyozásnál hozzáadunk minden változását az élek különböző megvalósításainak az időben [32].

És a fenti példa kiszámított eredményei [24, p. 344] az egyes szavak valószínűségével a felső sorokban:



7. ábra. A végeredmény.

A hasonló modelleknél a mintaadatok gyakran használnak kezdő és vége jelet, megkönnyítve a feldolgozást. Hasonló a beszédminták tanításánál is hasznos lehet, sőt ez segíthet a hangerő korrekt beállításához, vagy mint referencia fonéma. A *7. ábrán* furcsának hat az 1.3, illetve 1.7, mint valószínűség a *macska* szó esetében. A tanításhoz használt minták mennyisége sokat tud ezen javítani. Beszédazonosítás esetén is hasznos, ha ugyanaz a szöveg, nemcsak egy forrásból van meg, és javíthat a felismerés sikerességében, ha ugyanattól a beszélőtől is több mintánk van, azokkal a kulcs szavakkal, amik kritikusak (pl. jobb vagy bal).

5.5.1. Minta generálása

A tanulás visszaellenőrzéséhez gyakran szükséges mintákat generálni. A klasszikus esetben, az eredeti adatbázis előzetes felosztásával oldjuk meg a tanítást – tanitó / teszt felosztás, pl. 70/30 %-os arányban –, de relatív kis adatbázisok esetén ez nem mindig megoldható. Pl. az egy beszélőre optimalizált hangmintáknál is szükség lehet generált mintákra.

Az elgondolás itt, hogy mintát veszünk a minden időpontban generált zsetonból. Javítandó a sorosan generált zsetonok pontosságát, sugár keresés végezhetünk, amely folyamatosan követi a legvalószínűbb előzményeket, bármilyen hosszban. Vagyis az előre meghatározott szó sorrendekre nyilvántartja, hogy melyik a leginkább igéretes kombináció.

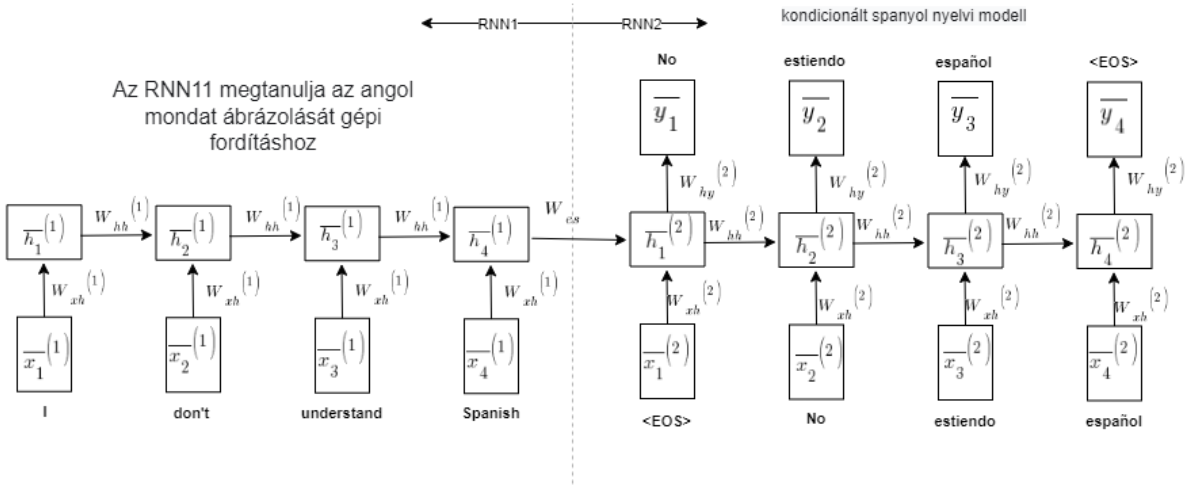
A karakter szintű visszacsatolt neurális hálózatnál pl. egy Shakespeare idézet elfogadható szintű felismeréséhez 50 iterációra volt szüksége ennek a modellnek. Beszéd esetén, inkább fonéma, mint karakter szint használható, de a módszer itt is működik [24, p. 346].

Az elsődleges célja a nyelvi-modellező RNN-nek, nem tetszőleges nyelvi sorozatok generálása, hanem hogy alapot biztosítson ahhoz, hogy a különböző, specifikus tartalmak felismerhetőek legyenek. Pl. a gépi fordítás tanítható forrás nyelv normál mintáival is. Minden esetben a kulcs, a bemeneti és kimeneti egységek számának korrekt meghatározása a visszacsatolt hálózatban, hogy mind a kimeneti hibák visszacsatolása, mind a súlyozások betanulása a felhasználás módjának megfelelő legyen.

5.5.2. A sorozatról-sorozatra tanulás és a gépi fordítás

A *sorozatról-sorozatra* (**sequence-to-sequence**) tanulás egyszerű megvalósítási lehetősége, amikor két RNN-t egymás után helyezünk ahogy a lenti ábrán is látható. Ez a specifikus példa angol-spanyol fordítást szemléltet [24], de ez csak egy konkrét felhasználás. Fontos észre venni a *8. ábrán*, hogy két RNN szerepel, saját súlyozásokkal. A $\bar{h}_4^{(1)}$ kimenet, az állandó hosszúságú kódolása a 4 szavas angol mondatnak. Bár a lenti megoldás is jól használható, inkább a később ismertetésre kerülő *long short-term memory (LSTM)* terjedt el a gépi fordítási feladatokra.

Mindkét oldalt meg kell tanítani a fordítás előtt a saját nyelvtani szabályaira ahhoz, hogy viszonylag pontos eredményt kaphassunk. A tanítás során párokat használunk mindkét oldalra. Mivel csak a célnyelv hálózatának van kimenete, ez kerül visszacsatolásra a kiinduló nyelv bemenetein keresztül.



8. ábra. 2 db RNN használata gépi fordítási feladathoz.

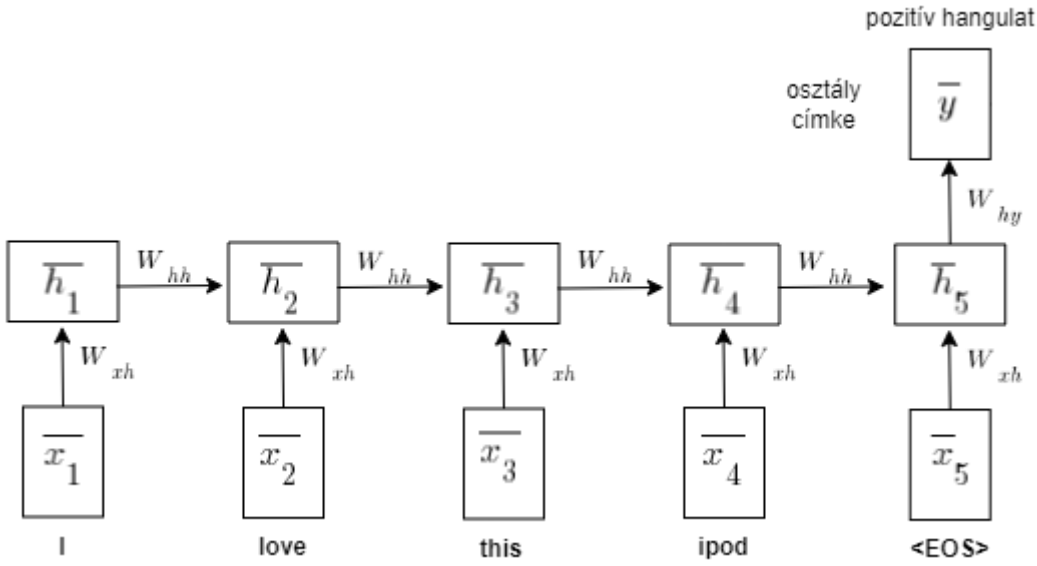
A *kérdés-felelet* (**QA**) típusú rendszerek hasonlóan működnek. A jelentős különbség – ahogy már korábban is utaltam rá – a tanításhoz használt adatbázis speciális összeállítása. Itt adott kérdések szerepelnek, amire adott készletből lehet válaszokat adni. Mindegy, hogy a gépi felismerés melyik oldalon szerepel – a kérdező vagy a válaszoló a gép -, fontos szerepe van az általános *Nem értem, kérem ismételje meg* típusú hibakezelő válasznak, amit szinte kötelező ezekbe a rendszerekbe beépíteni.

A másik típusú QA rendszer, amikor megengedünk általános kérdéseket is feltenni, nem korlátozzuk azokat. Ilyenkor bonyolultabb maga a felismerés is, de komoly szótár szükséges, komplett mondatok, bekezdések vagy szócikkek formájában. Ilyenkor valójában egy keresés történik, ahol ezt a lépést egy beszédfelismerés előzi meg, így a mesterséges intelligenciának kettős szerepe van. Ez nem összetévesztendő a sima Google kereséssel.

5.5.3. Mondat szintű osztályozás

Ebben a típusú osztályozási feladatban, minden mondat tanításhoz – vagy teszthez – használt. Ez a fajta osztályozás bonyolultabb, mint a teljes dokumentumra vonatkozó, főként a mondatok rövidsége miatt. A leggyakrabban azonban nagyon jól használható az osztályzásra, felismerésre – ami az osztályzás célja. A *9. ábrán* egy minta látható erre, ahol az érzelem detektálása a cél a mondat tartalmából.

Ahogy a mintából is látható, az egyes szavak érzelmi jelentése meghatározható, de a mondat szintű jelentés csak a szavak sorrendje alapján teljes. Figyelni kell a tartalmat megfordító, pl. tagadó szavakra [24, p. 352].

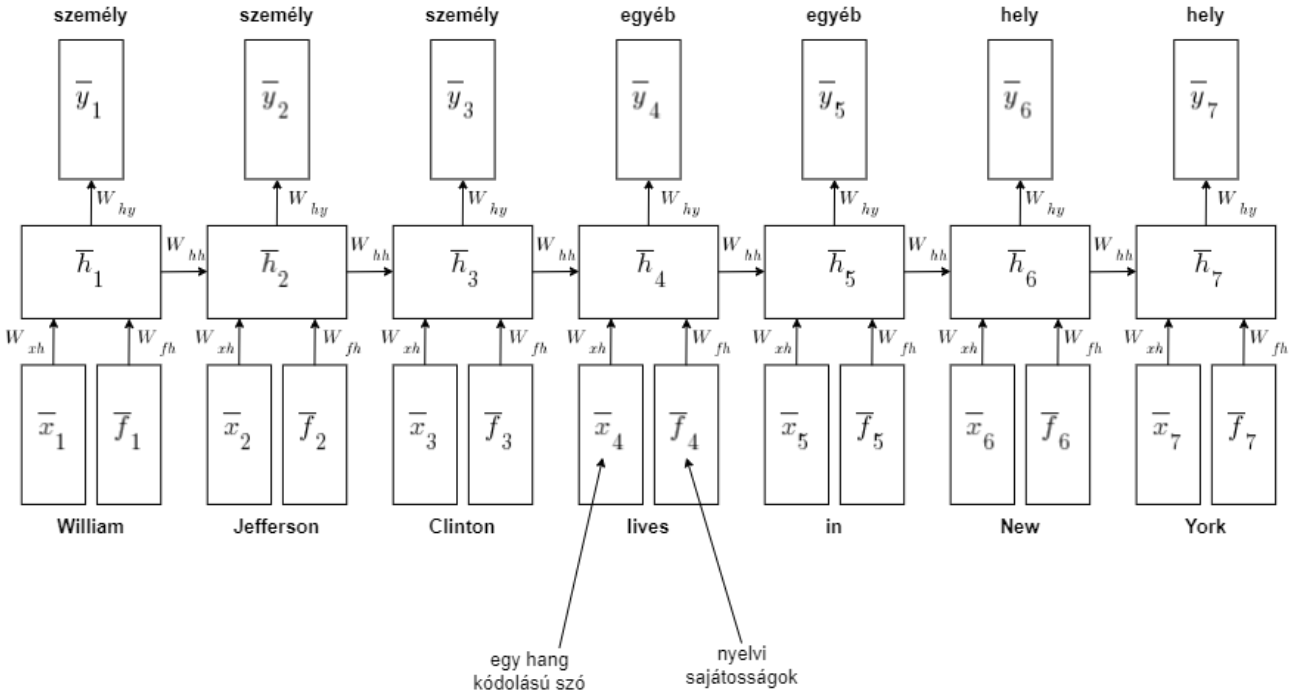


9. ábra. Mondat szintű osztályozás, érzelem analízisre.

5.5.4. A zseton (token) alapú osztályozás

A token vagy zseton alapú osztályozásnál a felismerendő, osztályozandó elemeket különböző cimkék alapján használjuk. Ilyen a *10. ábrán* látható mintában, pl. a *lives* szó, amely ragozott formában szerepel – nem a klasszikus szótári alakban, igy szükséges nyelvi jellemzők használata is az osztályozás során.

Ennél a módszernél a szóhoz kapcsolódó nyelvi jellemzők (nagybetű/kisbetű, pozíció a beszédben, helyesírás) sokkal fontosabbak, mint egy átlagos nyelvi modell, vagy gépi fordítás esetén. Az egyes szavakhoz rendelt kategóriák vagy címkék a tanítási során használtak, a teszt során nem ismertek. Látható különbség a korábbi modellekhez képest, hogy a rejtett réteg itt 2 bemenetet kap minden szóról és lehetőség van arra, hogy a nyelvi jellemzőkhöz más súlyozást rendeljünk, mint a szóazonosító részhez.



10. ábra. Zseton(token) alapú osztályozás, nyelvi funkciókkal.

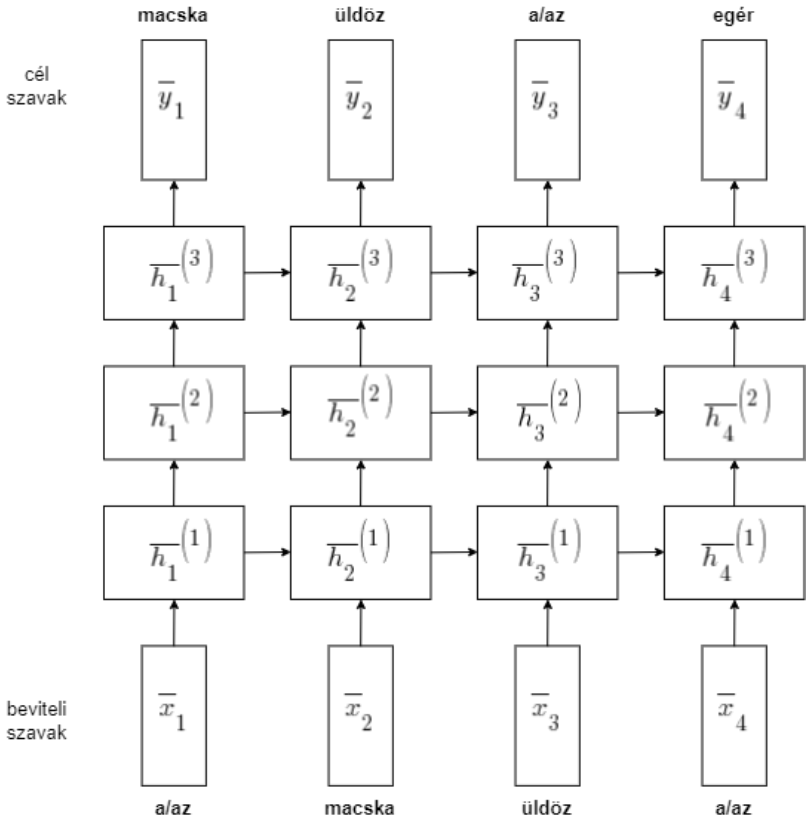
5.5.5. A többrétegű visszacsatolt neurális hálózatok

Az egyrétegű RNN, elsősorban a könnyebb megértést szolgálja, nagyon ritka, hogy gyakorlatban is használatba kerüljön. A valós problémákhoz általában többrétegű visszacsatolt neurális hálózatokat (Multilayer Recurrent Neural Network, MRNN) használunk. Az egyik megvalósítás a Long Short-Term Memory (LSTM) [24, p. 355].

A *11.ábra* egy többrétegű RNN-t ábrázol, ahol a bal oldal az alacsonyabb réteg, ami információval látja el a következő, magasabb réteget. A kapcsolatot a rétegek között a korábbi módszerrel lehet általánosítani:

$$\bar{h}_t = tanh\left(W_{xh}\bar{x}_t + W_{hh}\bar{h}_{t-1}\right) = \tanh W \begin{bmatrix} \bar{x}_t \\ \bar{h}_{t-1} \end{bmatrix} \tag{5.29}$$

Egy jóval nagyobb, $W = [W_{xh}, W_{hh}]$ mátrixot kell használnunk, megfelelő vektorokat. Az első rejtett réteg speciális, mert nemcsak a bemeneti réteg adatait kezeli a *t* időpillanatban, hanem a szomszédos rejtett réteg kimeneti adatát a *t-1* időpillanatból.



11. ábra. Egy multilayer(többrétegű) RNN.

Így a $W^{(k)}$ mátrixok mérete $p \times (d + p)$, de csak az első rétegre, ahol d a bemeneti $\bar{x}_t$ vektor mérete, és a p a rejtett réteg $\bar{h}_t$ vektorának mérete. Meg kell jegyezni, hogy d és p általában nem egyforma. A $k \geq 2$ esetre, a rejtett rétegek visszacsatolási értéke hasonlóan számolható a korábbiakkal:

$$\bar{h}_t^{(k)} = \tanh W^{(k)} \begin{bmatrix} \bar{h}_t^{(k-1)} \\ \bar{h}_{t-1}^{k} \end{bmatrix}$$

Ebben az esetben a $W^{(k)}$ mátrix mérete $p \times (p + p) = p \times 2p$. Az átalakítás az utolsó rejtett rétegről a kimeneti rétegre ugyanolyan, minta az egyszerű hálózatoknál. Praktikusan 2 vagy 3 rejtett réteg használatos a különböző megoldásoknál.

A visszacsatolt neurális hálózatok egyik problémája az eltűnő és hirtelen megnövekvő változások. Ez gyakori gond a neurális hálózatok frissítésénél, amikor a $W^{(k)}$ többszörözése nem stabil. Előfordulhat, hogy a változás eltűnik a visszacsatolás során, vagy hirtelen nagyon nagyra nő. Ezt úgy is magyarázhatjuk, hogy az RNN csak rövid sorozatokat tud jól kezelni, mivel ezek megkeverik a rejtett állapotokat. Ez a tulajdonság eredendően okoz jó rövidtávú, de gyenge hosszútávú memóriát.

Ennek a problémának a megoldását célozza a visszacsatolási egyenlet megváltoztatása a rejtett vektorra, az LSTM használatával.

Az LSTM egy olyan továbbfejlesztése a visszacsatolt hálózatoknak, ahol bevezetünk egy új, rejtett, p méretű vektort, amelyet $\bar{c}_t^{(k)}$ jelöléssel használunk. Ez a vektor a cella állapotára utal a mátrixban. Ez gyakorlatilag egy hosszútávú memória a cellára, amely megtartja legalább egy részét a korábbi rejtett állapotoknak, a részleges „felejtés" és „növelés" kombinációját használva. Ennek mikéntje a több szerző által is ismertetésre került, van ahol a szerző úgy próbálja magyarázni a $\bar{c}_t^{(k)}$ működését, hogy egy irodalmi mű darabjára illeszti. A vektor egyik p értékének előjele változhat a nyitó idézőjelnél és visszaállhat a záró idézőjelnél. Az előnye ennek a megközelítésnek, hogy az RNN képes modellezni a nyelv hosszútávú függőségeit, vagy akár speciális minták – pl. idézőjelek, vagy egyes kötőszavak a beszélt nyelvben – kiterjesztenek több zsetont.

Az így előállított mátrix $4p \times 2p$ méretű és bevezetünk 4, köztes, p méretű változót, $\bar{\imath}, \bar{f}, \bar{o}$ és $\bar{c}$, ahol az első három a bemenet, felejtés és kimenet változója, az utolsó pedig a cella állapotához rendelt.

$$\begin{bmatrix} \bar{\imath} \\ \bar{f} \\ \bar{o} \\ \bar{c} \end{bmatrix} = \begin{pmatrix} sigm \\ sigm \\ sigm \\ sigm \end{pmatrix} W^{(k)} \begin{bmatrix} \bar{h}_t^{(k-1)} \\ \bar{h}_{t-1}^{(k)} \end{bmatrix} \quad \text{[Az ideiglenes változók beállítása]}$$

$$\bar{c}_t^{(k)} = \bar{f} \odot \bar{c}_{t-1}^{(k)} + \bar{\imath} \odot \bar{c} \quad \text{[Szelektív felejtés és/vagy hozzáadás a hosszútávú memóriához]}$$

$$\bar{h}_t^{(k)} = \bar{o} \odot tanh\left(\bar{c}_t^{(k)}\right) \quad \text{[Szelektív szivárgása a hosszútávú memóriának a rejtett állapotba]}$$

Egy rövidebb matematikai átalakítás sorozat után jutunk el a rejtett rétegek állapotának egyenletéhez:

$$\bar{h}_t^{(k)} = \bar{o} \odot \bar{c}_t^{(k)} \tag{5.30}$$

Ahogy minden neurális hálózat esetén, a visszaterjesztés (**backpropagation**) algoritmus tanítási célokat szolgál. Meg kell említeni azt is, hogy sok – technikailag megvalósítható – sima RNN, sokkal jobb eredményt ad, amennyiben többrétegű LSTM-ként kerül végül megvalósításra.

Bár elsőre úgy tűnhet, hogy az LSTM módszer kizárólag szövegfeldolgozásnál használható, jó eredménnyel használható, a számunkra fontos beszédfelismerési feladatoknál is.

6. NLP Jupyter Notebook környezetben

A dolgozat záró részében Jupyter Notebook és a SPACY csomag segítségével bemutatom gyakorlatban is szövegfeldolgozást. A Jupyter Notebook nem kezeli dinamikusan a lefuttatott programsort, ezért a mellékletekben szerepelnek az adott kódok. A részegységeket képként mutatom be és mellé teszem a magyarázatot.

A SPACY egy természetes nyelv feldolgozására létrehozott csomag a Python nyelvhez. Open Source, tehát ingyenesen használható. Több, mint 72 nyelven elérhető és 24 nyelven 80 tanított feldolgozó *pipeline* érhető el. Komponensek a nevezett entitásokat felismeréséhez, a beszédrészek címkézése, függőségeket elemzése, mondatok szegmentálása, szöveg osztályozás, lemmatizálás, morfológiai elemzés, entitásokat összekapcsolása. [33] Eredetileg magyar nyelven nem érhető el, viszont a Szegedi Tudományi Egyetem jóvoltából létezik egy teljesen kompatibilis csomag melynek neve *HuSpaCy* [34]. Alább magyar nyelven mutatom az eredményeket.

6.1. Tokenizálás, címkézés és függőségek

A példában egy egyszerű mondat szerepel, melyet használni fogok. A programot és kimeneteit képként illesztettem be. Az egész program elérhető https://github.com/cpslabor-education/dipterv_textprocessing/tree/main/tokenizacio_jupyter linken.

A program sorok felett szerepel a magyarázat, viszont a program lefutásának magyarázatához bővebb információ szükséges.

A program futásához szükséges csomagok előhívása:

```
#szükséges csomagok meghívása
import spacy
import spacy_conll
import huspacy
from spacy import displacy
```

A magyar nyelvi modell betöltése az NLP definíciójaként, majd a megjelenítéshez pipeline létrehozása:

```
#nagy magyar nyelvi modell betöltése
nlp = huspacy.load()
#pipeline komponens létrehozása
nlp.add_pipe("conll_formatter", last=True)
```

A példa mondat: „Fazekas Zoltán Magyarországon vásárolta meg jó barátját Bellát a németjuhász kutyát a Kutyafarm Kft.től”. Az idézőjelben lévő mondatot a doc-ban tárolom, majd token-ekre bontom:

```
#a mondat felbontása tokenekké
doc = nlp("Fazekas Zoltán Magyarországon vásárolta meg jó barátját Bellát a
németjuhász kutyát a Kutyafarm Kft.től.")
for token in doc:
    print(token.text)
```

```
Fazekas
Zoltán
Magyarországon
vásárolta
meg
jó
barátját
Bellát
a
németjuhász
kutyát
a
Kutyafarm
Kft.től
.
```

12. ábra. Mondat token-ekké alakítása.

A mondat darabolásra került a *12. ábrán* látható, innentől a mondat elemei tokenként vannak nyilvántartva a programban. A token osztály különböző tulajdonságokkal rendelkezik, ezeket a tulajdonságokat string típusként tárolja:

```
#a token több, mint egy String és vannak tulajdonságai.
#a token típusú osztály olyan, mint egy String, de több attribútuma van

for token in doc:
    print(
        token.text
        + (14 - len(token.text)) * " "
        + token.text_with_ws
        + (14 - len(token.text_with_ws)) * " "
        + str(token.is_alpha)
        + (7 - len(str(token.is_alpha))) * " "
        + str(token.is_digit)
        + (7 - len(str(token.is_digit))) * " "
        + str(token.is_upper)
        + (7 - len(str(token.is_upper))) * " "
        + str(token.i)
    )
```

```
Fazekas       Fazekas       True   False  False  0
Zoltán        Zoltán        True   False  False  1
MagyarországonMagyarországon True   False  False  2
vásárolta     vásárolta     True   False  False  3
meg           meg           True   False  False  4
jó            jó            True   False  False  5
barátját      barátját      True   False  False  6
Bellát        Bellát        True   False  False  7
a             a             True   False  False  8
németjuhász   németjuhász   True   False  False  9
kutyát        kutyát        True   False  False  10
a             a             True   False  False  11
Kutyafarm     Kutyafarm     True   False  False  12
Kft.től       Kft.től       False  False  False  13
.             .             False  False  False  14
```

13. ábra. Tokenek és attribútumaik.

A token attribútumait lemmatizálhatjuk az adott token-re, mint a *13. ábrán*. A lemmatizáció egy a nyelvészetben használt kifejezés, amely egy adott szó – esetünkben token- ragozott formáinak csoportosításának folyamata azért, hogy egyetlen elemként elemezhető legyen. Tehát a szó szótári alakjával, azaz a lemmájával azonosítva legyen [35].

Módszert tekintve a szabály alapú lemmatizálás az, ahol véges állapotú automatákat lehet építeni nyelvészeti szabályrendszer alapaján, ami végi megy a token karakterein és így kerül meghatározásra a lemma és a morfológiai címkék [36]. Az Szegedi Tudományi Egyetem-en fejlesztett magyarlánc ezt a technikát alkalmazza. A neurális hálók esetében a tanítás során létrehozott szabályok alapján osztályozza azokat, amiket lehet alkalmazni a token-re, hogy eredményként a lemma létrejöjjön. A HuSpaCy neurális hálót alkalmaz ahol egy edit tree lemmatizer-t tartalmaz.

A példamondatot átalakítom, hogy az egyes szavak a SpaCy esetei legyenek, majd a SpaCy neurális hálón alapuló lemmatizálóját használom a s szavak szótári alakjának meghatározásához:

```
# tokenizálás
print("Tokenek:")
print([token.text for token in doc])

# lemmatizálás
print("\n\nLemmák:")
print([token.lemma_ for token in doc])

# POS tagging
print("\n\nPOS tagek:")
for token in doc:
    print_token_feautres(token)
```

```
Tokenek:
['Fazekas', 'Zoltán', 'Magyarországon', 'vásárolta', 'meg', 'jó', 'barátját', 'Bellát', 'a', 'németjuhász', 'kutyát', 'a', 'Kutyafarm', 'Kft.től', '.']


Lemmák:
['Fazekas', 'Zoltán', 'Magyarország', 'vásárol', 'meg', 'jó', 'barát', 'Bella', 'a', 'németjuhász', 'kutya', 'a', 'Kutyafarm', 'Kft.től', '.']


POS tagek:
Fazekas         Fazekas         PROPN  PROPN  nsubj        Xxxxx  True   False
Zoltán          Zoltán          PROPN  PROPN  flat:name    Xxxxx  True   False
Magyarországon  Magyarország    PROPN  PROPN  obl          Xxxxx  True   False
vásárolta       vásárol         VERB   VERB   ROOT         xxxx   True   False
meg             meg             PART   PART   compound:preverbxxx    True   True
jó              jó              ADJ    ADJ    amod:att     xx     True   True
barátját        barát           NOUN   NOUN   obj          xxxx   True   False
Bellát          Bella           PROPN  PROPN  obj          Xxxxx  True   False
a               a               DET    DET    det          x      True   True
németjuhász     németjuhász     ADJ    ADJ    amod:att     xxxx   True   False
kutyát          kutya           NOUN   NOUN   obj          xxxx   True   False
a               a               DET    DET    det          x      True   True
Kutyafarm       Kutyafarm       PROPN  PROPN  obl          Xxxxx  True   False
Kft.től         Kft.től         PROPN  PROPN  flat:name    Xxx.xxxFalse  False
.               .               PUNCT  PUNCT  punct        .      False  False
```

14. ábra. Lemmák.

Ismeretes, hogy a lemma keresése függ annak szófajától is, ezért meg kell vizsgálni az egyes szavak szófaját is. Figyelembe kell venni azt, hogy a szavak önállóan állnak kontextus nélkül, így a a szófaji meghatározás pontatlan lehet, a *14. ábra* ad magyarázatot:

A POS értékét is tudnunk kell, mely így ez:

- **ADJ:** melléknév
- **ADP:** adpozíció
- **ADV:** határozószó
- **AUX:** segéd
- **CCONJ:** koordináló konjunkció
- **DET:** meghatározó
- **INTJ:** közbeszólás
- **NOUN:** főnév
- **NUM:** szám
- **PARTICLE:** részecske
- **PRON:** névmás
- **PROPN:** tulajdonnév
- **PUNCT:** írásjelek
- **SCONJ:** alárendelő kötőszó
- **SYM:** szimbólum
- **VERB:** ige
- **X:** egyéb

Egy sor tag azt jelenti a nyolc oszlopban, hogy

TOKEN – LEMMA – POS - TAG – DEP – SHAPE - IS ALPHA - IS STOP

ahol a,

- **TOKEN** az eredeti szó a szövegben
- **LEMMA** a szó alapalakja
- **POS** az egyszerű címke
- **TAG** a címke, a részletes beszédrész címkéje
- **DEP** a szintaktikai függűség, azaz a token-ek közti kapcsolat
- **SHAPE** a szó alakja – nagybetű, kisbetú és számjagy
- **IS ALPHA** a karakter alfa-e
- **IS STOP** a token egy stoplista része-e, a nyelv leggyakoribb szava-e

A morfológia a nyelvészet azon ága, amely a szavak belső szerkezetét vizsgálja. A magyar nyelv alapvetően egy agglutináló nyelv, ami azt jelenti, hogy a szavaknak a szótövéhez végződések tartozhatnak, illetve akár előtag is társulhat egy adott szóhoz. A magyarban ezek a toldalékok fejezik ki a mondat szavainak egymáshoz való viszonyát, ellentétben például az angollal, ahol ezeket főként különálló, funkciószavak írják le. Az angolban így a magyarhoz képest nagyon kevés toldalalék létezik és azokat le lehet kezelni pár egyszerű szabállyal. Éppen ezért lesz nehezebb a magyar nyelvre egy morfológia elemzést végrehajtani, mert egy egyszerű előtag vagy toldalalék képes megváltoztatni a szó szerkezetét, jelentését. A *15. ábrán* figyelhető meg.

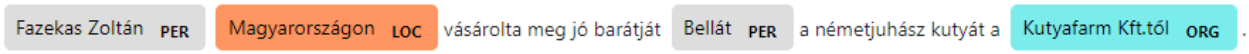
```
for word in doc:
 print('{} -> {}'.format(word.text, word.morph))
```

```
Fazekas -> Case=Nom|Number=Sing
Zoltán -> Case=Nom|Number=Sing
Magyarországon -> Case=Sup|Number=Sing
vásárolta -> Definite=Def|Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin|Voice=Act
meg ->
jó -> Case=Nom|Degree=Pos|Number=Sing
barátját -> Case=Acc|Number=Sing|Number[psor]=Sing|Person[psor]=3
Bellát -> Case=Acc|Number=Sing
a -> Definite=Def|PronType=Art
németjuhász -> Case=Nom|Degree=Pos|Number=Sing
kutyát -> Case=Acc|Number=Sing
a -> Definite=Def|PronType=Art
Kutyafarm -> Case=Nom|Number=Sing
Kft.től -> Case=Abl|Number=Sing
. ->
```

15. ábra. Morfológiai elemzés eredménye.

Megjelenítésre is létezik funkció, elsőként a címkézést mutatnám be a *16. ábrán*:

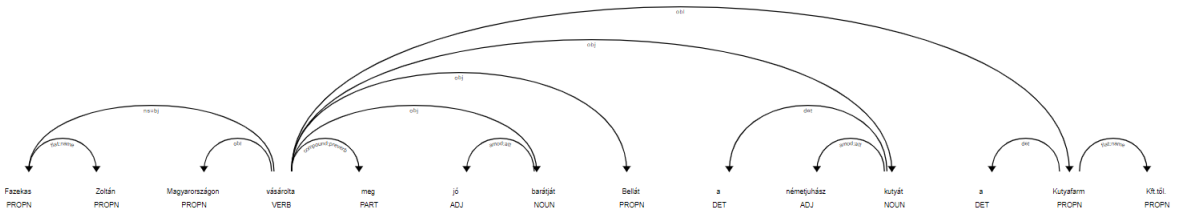
```
#megjelenítés
options = {"compact": True, "bg": "#09a3d5",
           "color": "white", "font": "Source Sans Pro"}
sentence_spans = list(doc.sents)
#displacy.serve(sentence_spans, style="dep")
#displacy.serve(doc, style="dep")
displacy.serve(doc, style="ent")d.text, word.morph))
```



16. ábra. Címkézés eredménye.

A címkézés jóvoltából A PER, mint Fazekas Zoltán és Bella személy felismerésre került, a LOC, mint Magyarország lokáció felismerésre került, és ORG, mint szervezet a Kutyafarm Kft. is felismert és entitásként címkézve lett.

Ha változtatom a „dep” opcióra, akkor a mondatat szintaktikai DEP token-ek közti kapcsolatot vizualizálja, mint a *17. ábrán* látható:



17. ábra. A tokenek jelentései és a köztük lévő összefüggés.

6.2. Szóbeágyazások

A szóbeágyazás arra nyújt megoldást, hogy nem független szóként lebontva kezeli a szöveget, hanem a szavak környezetét is vizsgálja, mert a tanítás a szavak környezetének használásával rendel egy-egy a szövegben előforduló szavakhoz egy sokdimenziós vektort [37]. Előrendelt, hogy a szóhoz rendelt vektornak hordoznia kell a jelentést is, tehát hasonló irányba mutató vektor hasonló jelentést kell reprezentálnia. Számos esetben vektorművelettel lehet kifejezni a jelentést.

6.2.1. Szóvektor a gyakorlatban

Léteznek előre elkészített modellek, amiket le lehet tölteni és használni lehet. Ezek általában az általános szóhasználatra vannak kihegyezve, azonban amennyiben egy olyan domain belül dolgozunk, ahol sok szakszó van ott egy saját modell tanítása jobb eredményt adhat. A program teljes egészében ezen a linken elérhető: https://github.com/cpslabor-education/dipterv_textprocessing/tree/main/szovektorok_jupyter.

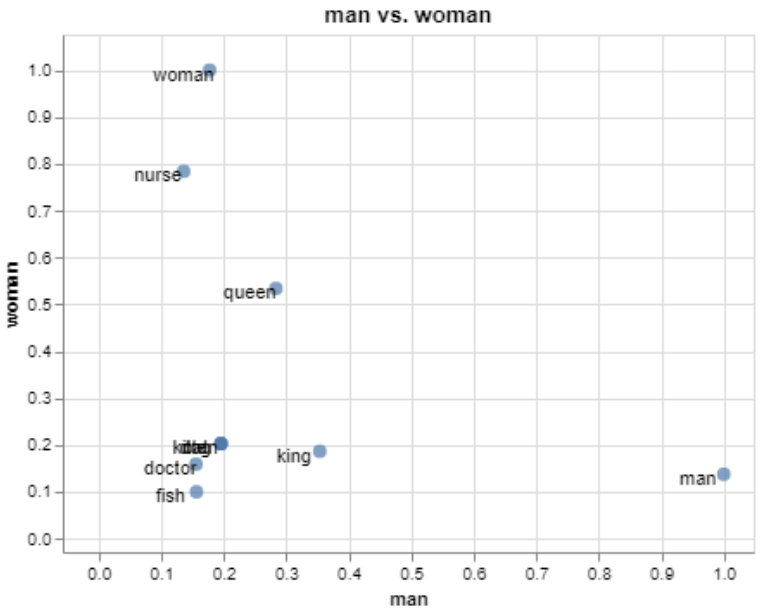
A program vizualizációt tartalmaz és megvizsgálom a Word2Vec vektorterét, a vektortér dimenzióredukálókat.

A whatlies csomag segíti a megjelenítést. Néhány angol szót betettem egy tömbbe és azt vizsgáltam, hogy a vektortérben a man-férfi és woman-nő szóhoz hogyan hasonulnak, a *18. ábrán* láthatóak az érdekes eredmények:

```
from whatlies import EmbeddingSet
from whatlies.language import SpacyLanguage
from whatlies.transformers import Pca, Umap, Tsne

lang = SpacyLanguage("en_core_web_md")
words = ["cat", "dog", "fish", "kitten", "man", "woman",
         "king", "queen", "doctor", "nurse"]

emb = EmbeddingSet(*[lang[w] for w in words])
emb.plot_interactive(x_axis=emb["man"], y_axis=emb["woman"])
```



18. ábra. Vektortérre példa.

Három vektortér redukálót is megnézek, hogy azok kimeneteit is megismerjem [38].

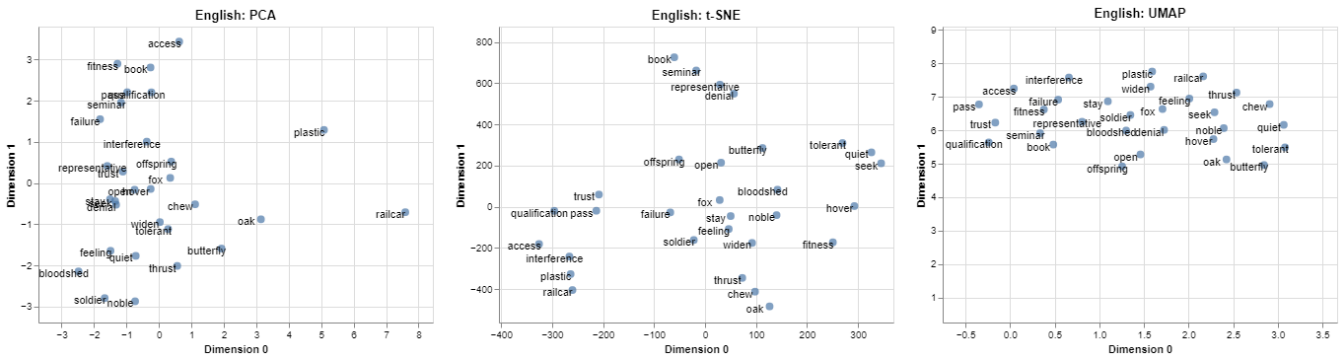
PCA: Principal Component Analysis – azonosítja az adatokhoz legközelebb eső hipersíkot, majd kivetíti az adatokat arra a hipersíkra és megtartja az adatkészlet variációknak nagy részét.

t-SNE: t-distributed Stochaistic Neighbour Embedding – egy nagy dimenziójú adatkészletet vesz fel és alacsony dimenziójú grafikonra redukálja, ami az eredeti információból sokat megőriz. Úgy, hogy minden adatpontnak helyet ad egy- két- vagy háromdimenziós térképen. Technikai klasztereket talál az adatokban, így biztosítva azt, hoghy a beágyazás megőrizze az adatok jelentését. Csökkenti a dimenziót, miközben igyekszik a hasonló példányokat közel tartani, a különböző példányokat pedig egymástól távol tartani. [39]

UMAP: Uniform Reduction for Data Visualisation - Az UMAP egy nemlineáris dimenziócsökkentési módszer, amely nagyon hatékony adatpontok klasztereinek vagy csoportjainak és relatív közelségeik megjelenítésére.

30 darab angol szót tettem egy tömbbe, ezeket vizsgálom:

```
["oak", "bloodshed", "railcar", "representative", "pass", "soldier",
"butterfly", "noble", "seminar", "interference", "plastic", "offspring",
"open", "chew", "seek", "failure", "qualification", "access", "tolerant",
"feeling", "trust", "fox", "quiet", "hover", "book", "widen",
"thrust", "denial", "stay", "fitness"]
```

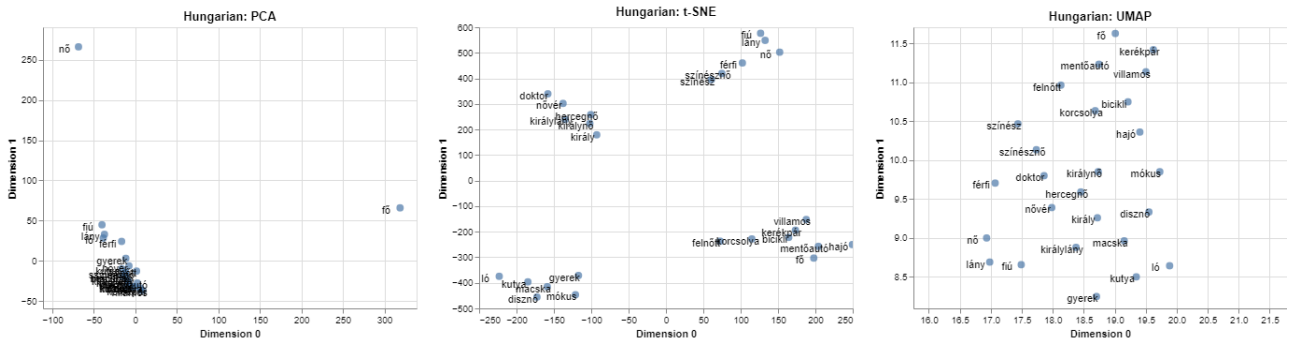


19. ábra. PCA, t-SNE, UMAP összehasonlítás.

A *19. ábrán* megfigyelhető, hogy a redukciós eljárással a mennyire válnak külön a szavak és ezáltal a tévesztés lehetősége is. Ránézésre a PCA-nál és a t-SNE-nél látszik keveredés és az UMAP jobban teljesít. Nyilván ebből konklúziót levonni nem érdemes.

Magyar nyelven is működik, 26 véletlenszerű szóra:

```
['disznó', 'ló', 'mókus', 'férfi', 'fiú', 'gyerek', 'korcsolya', 'bicikli',
'hajó', 'kerékpár', 'lány','nő', 'király', 'kutya', 'macska', 'hercegnő',
'királylány', 'királynő', 'doktor', 'nővér', 'színész', 'színésznő',
'felnőtt', "mentőautó", "villamos", "fő"]
```



20. ábra. Magyar nyelvre a redukció eredménye.

A *20. ábrán* szépen kirajzolódik, hogy itt még egyértelműbben az UMAP módszer hatásosabb.

6.2.1. Szóvektor modell tanítása Word2Vec módszerrel

Az eszköz a Gensim Word2Vec lesz, ennek segítségével lesz egy-egy modell tanítva. A korpusz beolvasása az első lépés. A korpusz nyelvészeti szakkifejezés, jelentése egy adott nyelv adott időpontban használt változatára vonatkozó szövegek összessége *[40]*. Egy olyan dokumentum lesz a célpont amit lezárt Stack Overflow kérdésekből állítottak össze, ahol csak a kérdések címei szerepelnek. Előfeldolgozás történt, mégpedig tokenizálva és normalizálva (speciális karakterek, írásjelek és számok eltávolításra kerültek) lett. Formátuma LineSentence – egy mondat (kérdés címe) soronként.

Alapvető információk a korpuszról:

```
print('A korpusz sorainak száma:', len(SO_corpus))
print('A korpusz teljes méret, i.e., az összes szó száma:',
recursive_len(SO_corpus))
print('Típus:', type(SO_corpus))
print('néhány példa sor:')
for j in range(5):
    i = random.randint(0, len(SO_corpus))
    print ('Sor:', i, ':', SO_corpus[i])
```

A korpusz sorainak száma: 202310
A korpusz teljes méret, i.e., az összes szó száma: 1110964

Típus: <class 'list'>
néhány példa sor:
Sor: 120317 : ['python', 'arrays', 'objects']
Sor: 144656 : ['rails', 'sum', 'values', 'array', 'hashes']
Sor: 159624 : ['errors', 'vs', 'think', 'recognising', 'classes']
Sor: 4323 : ['input', 'html', 'tag', 'parsing', 'properly', 'php']
Sor: 96401 : ['bus', 'error', 'using', 'gulp', 'watch', 'foundation', 'sass']

A Word2Vec modell építése CBOW módszerrel folytatódik. A tanítás egy rejtett réteggel történik, figyelembe veszi a környezetet és a tanult háló rejtett rétegét használja fel szóvektorként. Két fő fajtája:

CBOW: A környezet alapján tippeljük meg mi a köztes szó. Bemenet egy adott méretű környezet (pl. 2-2 szó előtte és utána), tanulandó a közöttük levő szó.

SKIP-GRAM: Adott szó alapján tippeljük meg a környezet szavait. Bemenet egy szó, tanulandó a környezete pl. 2-2 szó előtte és utána [41].

Bemenet és kimenet kódolása:

Bemenet one-hot encodinggal: teljes szótár méretének megfelelő vektor, ahol a bemeneti szónak megfelelő pozícióban 1 szerepel, minden más pozícióban 0.

Kimenet valószínűségekkel: szótár méretű vektor, a legnagyobb érték mutatja a háló által tippelt szó pozícióját.

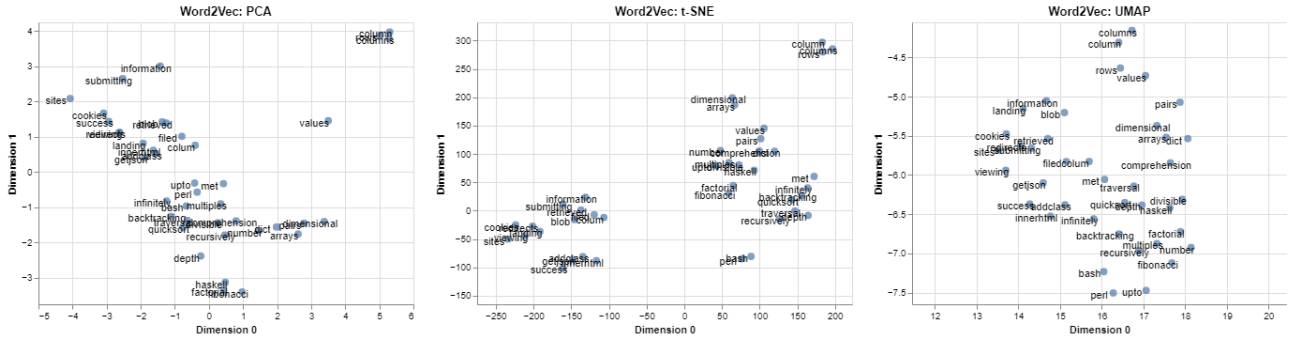
A modell tulajdonságai:
Word2Vec(vocab=8057, size=200, alpha=0.025)

Példák a *21. ábrán* a hasonlóságok láthatóak, oszlopok az általam megadott kulcsszavak a tíz sor pedig a hozzá leghasonlóbb szavakat mutatja:

	array	loop	javascript	python	digit	website	recursion	data	table	ajax
0	arrays	iteration	jquery	dict	digits	site	recursive	databse	tables	success
1	multidimensional	foreach	onchange	haskell	upto	websites	factorial	retrieved	row	getjson
2	associative	loops	vanilla	comprehension	numbers	viewing	iterative	values	rows	post
3	indexes	iterations	js	perl	number	pages	depth	information	column	callback
4	ints	looping	innerhtml	slicing	multiples	friendly	fibonacci	datas	pivot	request
5	dimensional	met	addclass	tuples	four	sites	traversal	retrive	records	axios
6	dimension	continue	attr	bunch	divisible	landing	quicksort	filed	columns	synchronous
7	nsmutablearray	infinite	textarea	dicts	consecutive	redirects	backtracking	rows	databse	submitting
8	pairs	increments	dom	bash	zeros	cookies	haskell	blob	alter	refreshing
9	hashtable	infinitely	getelementsbyclassname	recursively	five	seo	summation	retrieving	colum	onchange

21. ábra. Szavak hasonlósága a korpuszból kinyerve(Word2Vec).

Ha már rendelkezésre áll a vizualizáció lehetősége, így a DataFrame-ben megjelent szavakat a *22. ábrán.*



22. ábra. DataFrame(Word2Vec) szavai a redukált vektortérben.

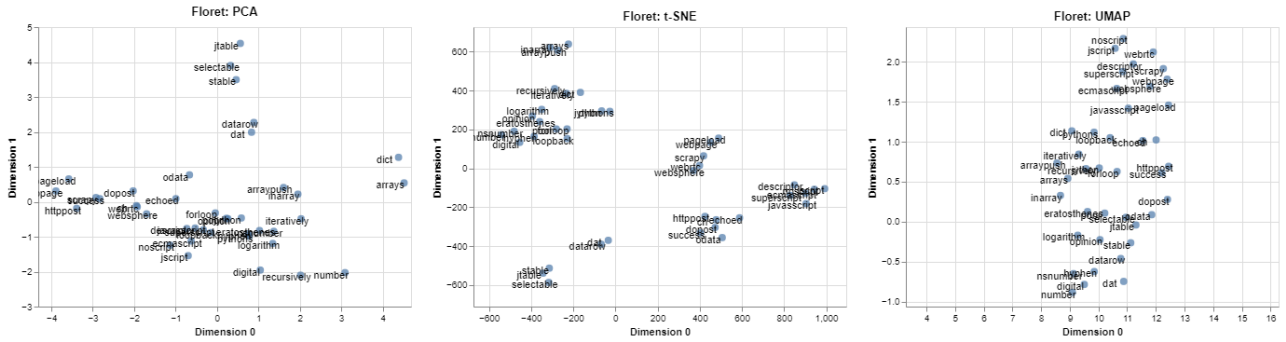
6.2.2. Szóvektor modell tanítása floret módszerrel

A floret a fastText kiterjesztett változata, amely képes bármilyen szóhoz szóreprezentációt készíteni egy kompakt vektortáblázatból. Egyesíti a fastText alszavai bármely szó beágyazásához és a Bloom beágyazások „kivonatoló trükkjét” [42].

Az előző módszer alapján elvégzem a modell tanítását és az alábbi *23. ábra* jön ki végeredménynek a leghasonlóbb szavakra és a *24. ábrán* a vektortérbeli elrendezésük.

	array	loop	javascript	python	digit	website	recursion	data	table	ajax
0	subarray	looped	javascripts	ipython	digits	site	recursive	datas	tableau	ch
1	arrays	loops	javasscript	pythons	digital	websites	recursively	databse	tablets	post
2	nsarray	loopback	ecmascript	jython	isdigit	sites	fibonacci	cdata	jtable	dopost
3	arraypush	met	noscript	pythonic	numbers	webpages	haskell	datarow	tablet	echoed
4	toarray	foreach	jscript	ironpython	number	scrapy	logarithm	dataset	datatable	succes
5	arrray	infinite	wscript	dict	numbered	scraper	depth	datastore	tables	httppost
6	subarrays	forloop	superscript	pythagorean	hyphen	webrtc	eratosthenes	dat	stable	onsubmit
7	arrary	looping	vbscript	tuples	nsdecimalnumber	websphere	opinion	jsondata	vtable	postman
8	inarray	poor	coffeescript	tuple	rownumber	webpage	maze	odata	datatables	pageload
9	nsmutablearray	iteratively	descriptor	dictonary	nsnumber	scraped	scenes	datatables	selectable	success

23. ábra. Szavak hasonlósága a korpuszból kinyerve(floret).



24. ábra. DataFrame(floret) szavai a redukált vektortérben.

A két módszert összehasonlítva, a feldolgozási időben látok különbséget. Mindkét esetben CPU-t használt a modell tanításra és abból is nyolc magot. A futási idő a Word2Vec esetében 56,6 másodperc alatt futott le a tanítás, a floret esetében 3 perc és 37 másodperc alatt. Számottevő különbség a példában nem jelentkezett a tanítási idő ellenére.

6.3. Szöveg, dokumentum beágyazások

A szavak szintje önmagában nagyon kevés információt tartalmaz. Esetenként szükségünk lehet további adatokra, ilyenkor szeretnénk nagyobb egységekre -például mondatokra vagy bekezdésekre- is egy-egy vektort előállítani. Erre néhány ötlet jöhet számításba:
A szövegben található szóvektorok közül válasszunk ki egy reprezentáns elemet vagy átlagoljuk ki a szóvektorokat és ez az átlag legyen a szövegszintű vektorunk vagy tanítsunk szövegszintű beágyazást.

Dokumentum szintű vektorok a SpaCy esetében viszonylag egyszerű, mert megoldhatjuk a szóvektorok átlagával. Ebben kissé gyanússá válik a dolog, hogy ha átlagoljuk a szavak jelentését akkor azoknak elveszik az értékük, tehát elveszik a nyelvi kifinomultság. Például egy szórenddel kifejezett jelentés kiemeléseket az átlag már nem különbözteti meg.

```
# Definájuk a Dokumentumunkat
doc = nlp_hu("villamossal megyek meccsre")

# Megkeressük a szavak vektorait
wv = []
for tk in doc:
    wv.append(tk.vector)

dv_1 = doc.vector # spaCy dokumentum vektor
dv_2 = np.mean([wv[0], wv[1], wv[2]], axis=0) # numpy-val kiszámítjuk a
szóvektorok átlagát
```

```
# Megnézzük, hogy a különbség 0-e, ha igen, akkor bizonyítottuk, hogy a
spaCy is így számol dokumentum vektort
all(v == 0 for v in dv_1 - dv_2)
```
→True

Ennek a kimente igaz, tehát bebizonyosodott, hogy a SpaCy is számol dokumentum vektort.

Mi történik, ha a példamondatban a meccs szó előtt kiemelem, hogy a meccs?

```
# Definájuk a Dokumentumunkat
doc = nlp_hu("villamossal megyek a meccsre")

# Megkeressük a szavak vektorait
wv = []
for tk in doc:
    wv.append(tk.vector)

dv_1 = doc.vector # spaCy dokumentum vektor
dv_2 = np.mean([wv[0], wv[1], wv[2]], axis=0) # numpy-val kiszámítjuk a
szóvektorok átlagát

# Megnézzük, hogy a különbség 0-e, ha igen, akkor bizonyítottuk, hogy a
spaCy is így számol dokumentum vektort
all(v == 0 for v in dv_1 - dv_2)
```
→False

Ennek a kimenete már hamis értéket mutat, tehát észrevehető az előbb említett hátrány. Viszont ha az alábbi példában ugyancsak jól teljesít:

```
doc1 = nlp_hu("villamossal megyek a térre")
doc2 = nlp_hu("a térre villamossal megyek")
print('Dokumentumok hasonlósága:', doc1.similarity(doc2))
```
→Dokumentumok hasonlósága: 0.9913338005910385

6.4. Dokumentumosztályozás szóbeágyazásra építve

A feladat megoldása során a Doc2Vec megoldást fogom használni az IMDB reviews adathalmazon. A *train_reviews* és a *test_reviews* ndarray objektumok, amelyek integereket tartalmazó listákból állnak. 25000 ilyen listát tartalmaz mindkét objektum. Egy-egy integereket tartalamzó lista egy-egy review. az integerek egyedi azonosítók, amelyek a szavakat kódolják. A *train-labels* és a *test_labels* szintén ndarray objektumok, melyek a reviewek címkéit tartalmazzák. Minden egyes reviewet pozitív(1) vagy negatív(0) kategóriába sorolandó.

Tensorflow.keras segítségével beállítom a címkéket majd a reviewsekre tanítom a modellt.

```
imdb = tf.keras.datasets.imdb
(train_reviews, train_labels), (test_reviews, test_labels) =
imdb.load_data()
train_reviews[0]
```

Szótárat kell készíteni azokkal a szavakkal amik reprezentálják az integerek dekódolását. Ehhez az imdb objektum *get_word_index()* metódusát használom, ami egy szótárat ad vissza, amelyben a szavak string alakja a kulcs és az integer alakja pedig a kulcs. A szótár elejéhez hozzá kell adni néhány tag-et, mint PAD, START, UNK, UNUSED.

```
vocab = imdb.get_word_index()
vocab = {k:(v + 3) for k, v in vocab.items()}
vocab["<PAD>"] = 0
vocab["<START>"] = 1
vocab["<UNK>"] = 2
vocab["<UNUSED>"] = 3
```

Érdemes egy inverz szótárt is létrehozni, ahol a szavak integer alakja a kulcs és a string alakja pedig az érték.

```
vocab_inv = dict([(value, key) for (key, value) in vocab.items()])
```

A decode_review() metódust definiálom, amely a review-ek integer reprezentációját string reprezentációvá alakítja.

```
def decode_review(review):
    return [vocab_inv.get(i, "?") for i in review]
decode_review(train_reviews[0])
```

A programkód teljes egészében megtalálható a kimenetekkel együtt itt: https://github.com/cpslabor-education/dipterv_textprocessing/tree/main/szovektorok_jupyter.

A dokumentumvektorok elkészítéséhez a gensim csomagot használom. Átalakítom a review-kat TaggedDocument objektumokká.

```
reviews = np.concatenate((train_reviews, test_reviews))
docs = [TaggedDocument(decode_review(review), [i]) for i, review in
enumerate(reviews)]
class Doc2VecCallback(CallbackAny2Vec):
    def __init__(self, epochs):
        self.prog_bar = tf.keras.utils.Progbar(epochs)
        self.epoch = 0
    def on_epoch_end(self, model):
        self.epoch += 1
        self.prog_bar.update(self.epoch)
```

A Doc2Vec osztály elkészíti a review-k dokumentum vektorait.

```
d2v_model = Doc2Vec(docs, dm=0, min_count=2, vector_size=100, hs=0,
negative=5, epochs=5,
                    callbacks=[Doc2VecCallback(5)], sample=0,
workers=multiprocessing.cpu_count())
```

A modellből ki kell nyerni a *train_reviews*-t és a *test_reviews* adatainak vektoros reprezentációját.

```
embdgs = d2v_model.docvecs.vectors_docs
train_embdgs, test_embdgs = np.split(embdgs, [25000])
train_embdgs[0]
```

Osztályozás Doc2Vec beágyazás alapján.

```
model_1 = tf.keras.Sequential()
model_1.add(tf.keras.layers.Dense(1, activation="sigmoid"))
model_1.compile(optimizer=tf.compat.v1.train.AdamOptimizer(0.01),
loss='binary_crossentropy',
              metrics=['accuracy'])
model_1.fit(train_embdgs, train_labels, batch_size=64, epochs=50,
shuffle=True)
```

Melynek eredménye:

```
782/782 [==============================] - 4s 4ms/step - loss: 0.3399 - accuracy: 0.8523

[0.3398783206939697, 0.8523200154304504]
```

Egy fokkal összetettebb neuronháló létrehozása.

```
model_2 = tf.keras.Sequential()
model_2.add(tf.keras.layers.Dense(50, activation="relu", input_shape=(100,
)))
model_2.add(tf.keras.layers.Dropout(0.3))
model_2.add(tf.keras.layers.Dense(50, activation="relu"))
model_2.add(tf.keras.layers.Dropout(0.2))
model_2.add(tf.keras.layers.Dense(50, activation="relu"))
model_2.add(tf.keras.layers.Dense(1, activation="sigmoid"))
model_2.compile(optimizer=tf.compat.v1.train.AdamOptimizer(0.01),
loss='binary_crossentropy',
              metrics=['accuracy'])
model_2.fit(train_embdgs, train_labels, batch_size=64, epochs=50,
shuffle=True)
```

Melynek eredménye:

```
782/782 [==============================] - 5s 6ms/step - loss: 0.3346 - accuracy: 0.8552

[0.3346216678619385, 0.8551999926567078]
```

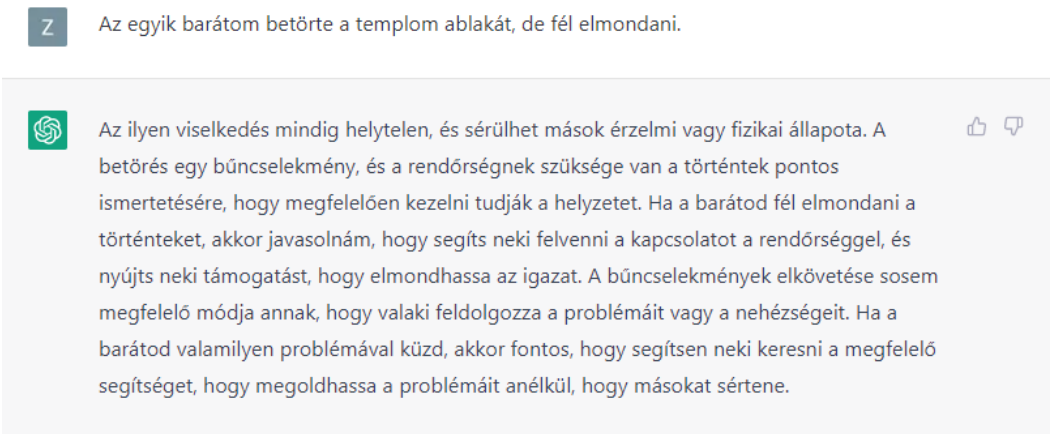
Az adathalmazon a tanítás sikeresen végrehajtódott. Ezután a modell képessé válik például szöveges osztályozásra, egy előre jelző rendszer létrehozásához, hogy milyen a film kritika hangulata vagy kérdés-válasz rendszer vagy mesterséges dokumentum generálásra.

7. Összegzés

A nyelvi feldolgozás, mint egy mesterséges intelligenciára épülő tevékenység. Mára az egyre jobban fejlődő algoritmusok és egyre nagyobb számítási kapacitással rendelkező számítógépek miatt a nyelvészeti megoldások tárháza hatalmas. Gondoljunk bele, hogy egy szótár mára már nem is szótár, hanem egy teljes fordító, ami képes a fordítandó nyelv kontextusait felismerni és a cél nyelvre közel tökéletesen visszaadni a jelentés tartalmát gyakorlatilag a világ összes nyelvén egy gombnyomásnyi idő alatt. A saját nyelven beszélő chatbotok ügyfélszolgálatokon a gyakran ismert és feltett kérdésekre segítenek az embereknek eligazodni az ügyük megoldására. Pár éve még egy esetlen, néhány száz kifejezést ismerő chatbot szórakoztatta az érdeklődőket, ma pedig az OpenAI chat GPT gyakorlatilag már-már azt az érzést kelti, hogy valóban egy gondolkodó és nagy tudással rendelkező élőlénnyel diskurál, a *25. ábrán* példa rá.

A diplomamunka utolsó részében a programozás Python nyelven történt, viszont Windows környezetben. A programnyelvről tudni illik, hogy kissé Linux orientált. A programozást lehetett volna egy virtuális gépen létrehozott Linux alatt megtenni és akkor sok probléma nem merült volna fel. Ennek a megoldására azért nem került sor, mert az elvégzett feladatok számításigényesek és az eszközöm erőforrásai nem támogattak. Ennek ellenére a vektortérben való vizualizáció segít az elméleti részt jobban megérteni vagy egy összetett neurális hálón való tanítást is lehetett érzékelni, hogy szükséges-e és az eltelt idő az arányos-e azzal, amit nyújtani tud.

A dolgozatot néhány év múlva érdemes lehet elővenni és folytatni, vagy csak átfutni, mert majdnem biztos vagyok benne, hogy összetettebb algoritmusok miatt a nyelvi feldolgozás még ennél is több lehetőséget fog tartogatni.



25. ábra. OpenAI chat *[43]*.

Irodalomjegyzék

[1] Magyar Telekom Nyrt., „Chat-segítség Telekom mobil Gyakori Kérdések,” Magyar Telekom Nyrt., 03 October 2022. [Online]. Available: https://www.telekom.hu/lakossagi/ugyintezes/gyakori-kerdesek/chat_segitseg. [Hozzáférés dátuma: 03 October 2022].

[2] N. Albayrak, A. Ozdemir és E. Zeydan, „An Artificial Intelligence Enabled Data Analytics Platform for Digital Advertisement,” in *22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, Paris, France, 2019.

[3] T. M. Mitchell, Machine Learning, New York, NY: McGraw-Hill, Inc., 1997.

[4] G. T. N. S. Fekete István, Bevezetés a mesterséges Intelligenciába, Budapest: LSI Oktatóközpont, 1990.

[5] G. Barcza, „Gépi tanulás a számításos kvantumkémiában,” *MAGYAR KÉMIAI FOLYÓIRAT-KÉMIAI KÖZLEMÉNYEK (1997-),* %1. kötet128(1), pp. 29-34., 2022.

[6] B. Megyeri, Természetes nyelvű interfész alapú ajánló rendszer fejlesztése a Google Dialogflow rendszer keretében., Miskolc: Miskolci Egyetem, 2019.

[7] S. BRIN, „ALGORITMUSOK VEZÉRELTE SORSOK,” *Szabad Piac,* %1. kötet1, pp. 18-22, 2021.

[8] S. V. Rózsa, Szövegfeldolgozás gépi tanulási módszerekkel, Budapest, Hungary: ELTE Természettudományi Kar, 2018.

[9] M. Borza, Mesterséges neurális hálózatok matematikai alapjai, Budapest: ELTE TTK Számítógép tudományz Tanszék, 2019.

[10] S. József, Modellek a geoinformatikában, Szeged: Szegedi Tudományegyetem; Debreceni Egyetem; Pécsi Tudományegyetem, 2013.

[11] K. O. a. R. Nash, An Introduction to Convolutional Neural Networks., ArXiv e-prints, 2015.

[12] S. F. a. F. G. Alex Graves, „Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks,” In Proceedings of the International Conference on Machine, 2006, pp. 369-376.

[13] A.-r. M. a. G. E. H. Alex Graves, Speech recognition with deep recurrent neural networks, CoRR, abs/1303.5778, 2013.

[14] M. P. P. B. S. Z. C. L. Y. B. a. A. C. C. Ying Zhang, Towards end-to-end speech recognition with deep convolutional neural networks, CoRR, abs/1701.02720, 2017.

[15] Google Inc., „Google Translator,” Google, 2022. [Online]. Available: https://translate.google.com/. [Hozzáférés dátuma: 5 oktober 2022].

[16] H. Ghasemi és M. Hashemian, „A Comparative Study of Google Translate Translations: An Error Analysis of English-to-Persian and Persian-to-English Translations,” *Englis Language Teaching by Canadian Center of Science and Education,* %1. kötet9, %1. szám3, pp. 13-17, 2016.

[17] Y. K. Y. D. J. S. a. A. M. Guillaume Klein, Opennmt: Open-source toolkit for neural machine translation, CoRR, 2017.

[18] K. C. a. Y. B. Dzmitry Bahdanau, Neural machine translation by jointly learning to align and translate., Computing Research Repository, 2014.

[19] T. K. E. G. L. E. W. K. M. S. P. B. Karl Moritz Hermann, „Teaching machines to read and comprehend,” *Advances in Neural Information Processing Systems,* %1. kötet28, pp. 1693-1701, 2015.

[20] A. K. Varga, „Gépi beszédfelismerés,” in *Fiatal Műszakiak Tudományos XVII. Ülésszaka*, Kolozsvár, 2012.

[21] L. J. Dupont S., „Audio-Visual Speech Modeling for Continuous Speech Recognition,” *IEEE Transactions on Multimedia,* %1. kötet2, %1. szám3, 2000.

[22] K. Vicsi, „Beszédadatbázisok a gépi beszédfelismerés segítésére,” *Híradástechnika,* %1. kötet1, pp. 5-13, 2001.

[23] P. Mihajlik, Spontán magyar nyelvű beszéd gépi felismerése nyelvspecifikus szabályok nélkül, Budapest: BME-VIK Villamosmérnöki Doktori Iskola, 2010.

[24] C. C. Aggarwal, Machine Learning for Text, Cham, Switzerlan: Springer International Publishing AG, 2018.

[25] K. C. G. C. a. J. D. Tomas Mikolov, Efficient estimation of word representations in vector space, 2013.

[26] „Glove: Global vectors for word representation,” *Empirical Methods in Natural Language,* p. 1532–1543, 2014.

[27] O. a. Y.Goldberg, „Neural word embedding as implicit matrix factorization,” *Advances in neural information processing systems,* %1. kötet27, 2014.

[28] K. L. a. C. Burgres, „Producing high-dimensional semantic spaces from lexical co-occurence,” *Behavior Research Methods,, Imstruments, and Computers,* %1. kötet28, %1. szám2, pp. 203-208, 1996.

[29] J. B. a. J.Levy, „Extracting semantic representations from word co-occurance statistic: A computational study,” *Behavior Research Methods,* %1. kötet39, %1. szám3, pp. 510-526, 2007.

[30] Y. a. I. O.Levy, „Improving distributional similarity with lessons learned from word embeddings,” *Transactions of the Association for Computational Linguistic,* %1. kötet3, pp. 211-225, 2015.

[31] I. S. K. C. G. S. C. a. J. D. Tomas Mikolov, „Distributed representations of words and phrases and their compositionality,” *Advances in Neural Information Processing Systems,* %1. kötet26, %1. számCurran Associates, Inc, p. 3111–3119, 2013.

[32] Y. A. I.Goodfellow, Deep Learning, MIT, 2016.

[33] I. M. Matthew Honnibal, „Industrial-Strength Natural Language Processing IN PYTHON,” ExplosionAI GmbH, 2022. [Online]. Available: https://spacy.io/. [Hozzáférés dátuma: 18 november 2022].

[34] G. Orosz, Z. Szántó, P. Berkecz, G. Szabó és R. Farkas, „HuSpaCy: an industrial-strength Hungarian natural language processing toolkit,” in *XVIII. Magyar Számítógépes Nyelvészeti Konferencia*, Szeged, 2022.

[35] gobertpartners.com, „Mit jelent lemmatizálni?,” 2022. [Online]. Available: https://gobertpartners.com/what-does-lemmatize-mean. [Hozzáférés dátuma: 11 december 2022].

[36] J. Csima és K. Friedl, „Nyelvek ´es automat´ak,” augusztus 2013. [Online]. Available: http://www.cs.bme.hu/~friedl/nyau/jegyzet-13.pdf. [Hozzáférés dátuma: 11 december 2022].

[37] Z. Kmetty, „Szóbeágyazási vektortérmodellek társadalomtudományi alkalmazása,” *STATISZTIKAI SZEMLE,* %1. kötet100, %1. szám2, pp. 105-136, 2022.

[38] S. Sivarajah, „https://towardsdatascience.com/,” 31 május 2020. [Online]. Available: https://towardsdatascience.com/dimensionality-reduction-for-data-visualization-pca-vs-tsne-vs-umap-be4aa7b1cb29. [Hozzáférés dátuma: 11 12 2022].

[39] v. d. L. Maaten és G. Hinton, „Visualizing Data using t-SNE,” in *Journal of Machine Learning Research 9*, 2008.

[40] Wikipédia, „Korpusz”.

[41] S. Doshi, „Towards Data Science,” 16 március 2019. [Online]. Available: https://towardsdatascience.com/skip-gram-nlp-context-words-prediction-algorithm-5bbf34f84e0c. [Hozzáférés dátuma: 11 december 2022].

[42] „Github,” [Online]. Available: https://github.com/explosion/floret/tree/main/python. [Hozzáférés dátuma: 11 december 2022].

[43] OpenAI, „ChatGPT,” 2022. [Online]. Available: https://chat.openai.com/chat. [Hozzáférés dátuma: 13 december 2022].

[44] Google, „Contexts,” Google Inc, 30 September 2022. [Online]. Available: https://cloud.google.com/dialogflow/es/docs/contexts-overview. [Hozzáférés dátuma: 3 October 2022].

Ábrajegyzék

Mellékletek

A program forráskódjai: https://github.com/cpslabor-education/dipterv_textprocessing.