

Kondor Martin

MAGYAR NYELVŰ ÁLHÍREK FELISMERÉSE MÉLY NEURÁLIS HÁLÓZATTAL

Témavezető: Dr. Jaskó Szilárd

Műszaki Informatikai Kar, Alkalmazott Informatikai Tanszék

Pannon Egyetem
Intézményi Tudományos Diákköri Konferencia
2023



Absztrakt

*Készítette: Kondor Martin,
Mérnök-informatikus BSc,
Műszaki Informatikai Kar,
Alkalmazott Informatikai Tanszék*

Témavezető: Dr. Jaskó Szilárd

A közösségi média és az okoseszközök terjedésével az emberek napról napra egyre több információhoz juthatnak hozzá. Nem csak az információ terjedése, de a terjedésének sebessége is drasztikusan megnőtt az utóbbi évtizedekben, ez a felhasználóknak köszönhető, akik megosztásaikkal és továbbításaikkal olyanokhoz is eljuttatják az éppen adott híreket, akik nem is ismerik az információ forrását. Ennek mellékhatása az, hogy nemcsak a valós hírek, hanem a direkt figyelemfelkeltési és félrevezetési céllal megírt hírek (álhírek) is gyorsabban terjednek. A mesterséges intelligencia segíthet a felhasználóknak abban, hogy ezeket az ún. álhíreket könnyebben felismerhessék és akár teljesen meg is szüntethessék az ezeknek való kitettségüket.

A nemzetközi irodalomban sok tudományos publikáció [1] foglalkozik az angol nyelvű álhírek felismerésével a mesterséges intelligencia segítségével, ezek többnyire döntési fákkal és különböző struktúrájú neurális hálózatokkal működnek, elég magas ~90%-os beazonosítási pontossággal, többnyire 10 000 mondat feletti mintaszámmal dolgoznak.

A magyar nyelvű álhírek automatikus beazonosítására is van a magyar szakirodalomban példa [2], ahol egy olyan gépi tanulási modellt alkalmaznak, ami 80%-os eredményességgel képes megfelelő csoportba sorolni az ál- és valódi híreket. Döntési fát használnak benne és a tanuló halmaz összesen 80 hírből (1720 mondatból) áll. Ezen kívül szükséges megemlíteni, hogy a hazai tanulmány a direkt szórákoztatásra megírt álhírek felismerését hajtja végre.

A munkámban bemutatott módszer a nemzetközi irodalomhoz [4] hasonlóan mély neurális hálókat használ, és felhasználja az éppen elérhető legfrissebb magyar szövegfeldolgozási technikákat a lehető legmagasabb pontosság elérése érdekében, ezen felül csak a hírek címei alapján próbál meg osztályozni. Az adatgyűjtés érdekében készítettem egy kérdőívet, aminek segítségével bárki jelenthet álhíreket. A kutatás során a magam által és a fent említett kérdőív segítségével gyűjtött híreket használtam fel, melyekből egy 1500 ál-, 1500 valós- hírcímekből álló adatbázist építettem ki, ezeket a híreket kézzel és automatikus módon is szűröm. A tanulási folyamat eredményessége érdekében különböző szövegfeldolgozási technikákat próbálok ki az adatok neurális hálózatra való felkészítéséhez.

A témában már volt magyar nyelvű álhírek azonosítását magába foglaló tanulmány [5], ám az én módszerem egy különböző struktúrájú mély neurális hálózatot használ, a saját magam és a kérdőív által gyűjtött hírcímek alapján dolgozik, egyedi szövegfeldolgozási módszereket használ és a lehető legpontosabb azonosításra koncentrál.

Kulcsszavak: mesterséges intelligencia, neurális hálózatok, mély tanulás, álhír, hír, magyar nyelvi feldolgozás.

Köszönetnyilvánítás

Szeretnék köszönetet mondani Dr. Jaskó Szilárd témavezetőmnek, aki végig felügyelte a munkámat, mindig válaszolt az esetlegesen felmerülő kérdéseimre, rendelkezésemre bocsátott erőforrásokat a böngésző bővítmény megvalósításához, és segített felkeresni különböző szervezeteket az adathalmaz ellenőrzése érdekében.

Tartalomjegyzék

1. Bevezetés.....	1
2. Mély tanulás alapjai.....	3
2.1. Matematikai alapok.....	3
2.1.1. Skalár.....	3
2.1.2. Vektor.....	4
2.1.3. Mátrix.....	4
2.2. Alapvető fogalmak.....	5
2.2.1. Mesterséges intelligencia fogalma.....	5
2.2.2. Gépi tanulás fogalma.....	6
2.2.3. Mély tanulás fogalma.....	6
3. Szakirodalmi áttekintés.....	7
3.1. Vincze Veronika – Kacsa vagy nem kacsa? Magyar nyelvű álhírek automatikus azonosítása .	7
3.2. Veszelszki Ágnes - Az álhírek extra- és intralingvális jellemzői	8
3.3. Thota A., Tilak P., Ahluwalia S., Lohia N. - Fake News Detection: A Deep Learning Approach	8
3.4. Godbole V., Dahl G. E., Gilmer J., Shallue C. J., Nado Z. - Deep Learning Tuning Playbook ...	10
4. Felhasznált technológiák.....	11
4.1. Szoftvereszközök	11
4.2. Szoftverkönyvtárak.....	13
4.3. Hardvereszközök	13
5. Adatgyűjtés és feldolgozás.....	15
5.1. Adatok gyűjtése.....	15
5.1.1. Sajátkezű gyűjtés.....	16
5.1.2. Automatizált gyűjtés	16
5.1.3. Kérdőíves gyűjtés	17
5.1.4. Összefoglalás.....	17
5.2. Adatok szűrése	18
5.2.1. Szoftveres szűrés.....	18
5.2.2. Kézi szűrés.....	19
5.2.3. Szakértői szűrés.....	19
5.3. Adathalmaz elemzés	19
5.4. Adatfeldolgozás.....	22
5.4.1. Adatfeldolgozási folyamat	23
5.4.2. Szövegfeldolgozás	26
5.4.3. Szövegfeldolgozási módszer megállapítása	33
6. Modell fejlesztés és kiértékelés.....	34
6.1. Modelltípusok áttekintése	34
6.1.1. Teljesen kapcsolt neurális hálózat	34
6.1.2. Visszacsatolt neurális hálózat	35
6.1.3. Konvolúciós neurális hálózat	35
6.2. A modell fejlesztése.....	35
6.2.1. Előzetes mérések	36

6.2.2. Lehetőségek a modell javítására.....	37
6.2.3. A modell módosítása	37
6.3. A modell finomítása.....	44
6.3.1. Adatgyűjtés	44
6.3.2. Adatszűrés.....	44
6.3.3. A modell tovább tanítása	46
6.4. A modell megállapítása.....	46
7. Modellek tesztelése.....	50
7.1. Teszthalmaz módszer	50
7.2. K-szoros keresztvalidációs módszer	51
7.3. Egyéb tesztek a tanítás után.....	52
7.3.1. Pontosság táblázatba foglalása	52
7.3.2. Variancia teszt.....	53
7.3.3. Zavarmátrix	55
7.4. Utótesztelés	57
8. Alkalmazhatóság és további feladatok	59
8.1. Alkalmazhatóság	59
8.1.1. Böngésző bővítmény.....	59
8.1.2. Tűrőhatár	62
8.1.3. Automatikus ellenőrzés	63
8.2. További feladatok.....	63
8.2.1. További adatgyűjtés	63
8.2.2. Hírszövegek felhasználása	64
8.2.3. Képek felhasználása	64
8.2.4. Időtállóság.....	64
9. Összefoglalás	65
10. Magyarításjegyzék	66
11. Ábrajegyzék	67
12. Táblázatjegyzék	69
13. Ajánlott irodalom	70
14. Irodalomjegyzék.....	71

1. Bevezetés

Az utóbbi években a közösségi média és az Internet népszerűségének köszönhetően az álhírek terjedése egyre nyomasztóbb problémává vált. Probléma, mert az Internetet böngésző személyek döntéseit és véleményeit befolyásolhatják az olvasott információk. Ez a probléma nem korlátozódik egy országra vagy nyelvre, hanem globális kérdés. Magyarországon sem más a helyzet, az Interneten egyre több hamis hír terjed. Forrásukat tekintve legtöbbször az álhírek olyan weboldalakról származnak, ahol az oldal üzemeltetői a nagyobb forgalom elérése érdekében terjesztik őket. Az álhíreket a tartalmuk alapján csoportosíthatjuk az alábbi kategóriákba:

- Áltudományos tartalmú
- Összeesküvés-elméletet taglaló
- Kattintásvadász célzatú
- Kitalált hírt közlő

Van, hogy a felsorolt kategóriák közül egy álhíre több kategória is igaz, és van, hogy egy hír olyan tartalommal rendelkezik, ami álhírekre jellemző, de valós információk is találhatóak benne. Általában az álhírekre jellemző bármilyen tartalom jelenléte egy hírben azt eredményezi, hogy álhírként kategorizáljuk az adott hírt.

Az álhírek jelenlegi formájukban az Interneten keresztül nagyon gyorsan képesek terjedni, azonban már az ókori Rómában is jelen voltak [73]. Az egyik első álhír terjesztésként feljegyzett esemény például az volt, amikor Augustus császár egy a római lakosságot félreinformáló kampányt szervezett, melynek célja Marcus Antonius hírnevének aláásása volt, ahhoz, hogy az álhírek a lehető legtöbb emberhez eljuthassanak, a császár pénzérmékre íratott rövid és megalázó állításokat Marcus Antoniusról [18].

Az álhírek megkülönböztetése a valós hírektől nem egy újkeletű próbálkozás, már az első nagyobb tömegeknek szánt hírközlők megjelenése óta vannak, akik arra törekednek, hogy megállítsák a hamis hírek terjedését. Magyarországon a leggyakoribb módszer az álhírek felismerésére a hírt olvasó személy tudásán és a kritikus gondolkodási képességén alapul. Ilyen álhírekre alkalmazható felismerési módszerek, például [ifla.org]:

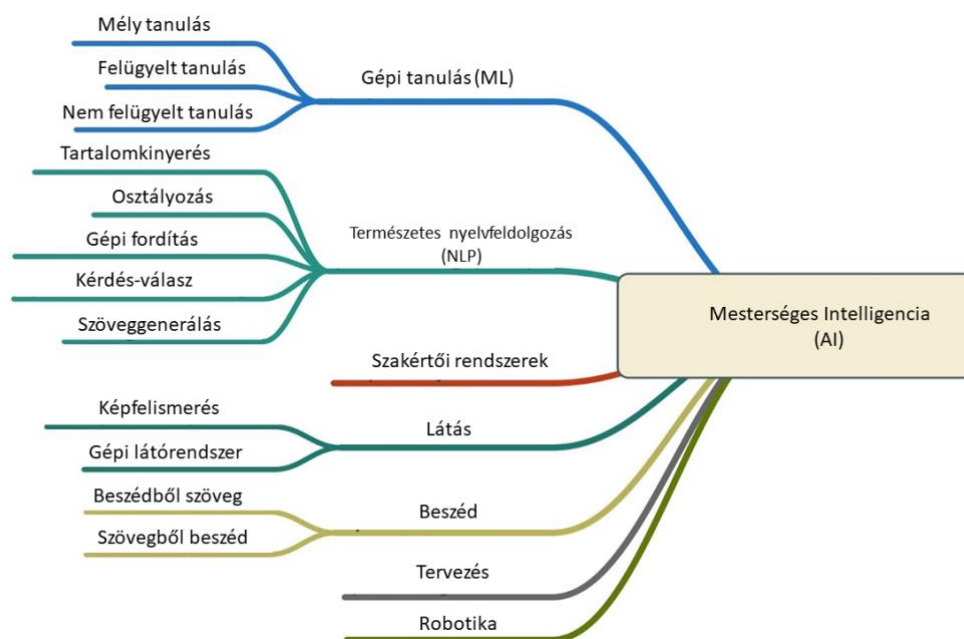
- Forrásmérlegelés
- Túlzó címsorokról való felismerés
- Szerző ellenőrzése
- Szakértők megkérdezése

Tekintve azt, hogy az emberek többségét nem oktatják az álhírek felismeréséről, érdekesnek tartottam az álhírek automatikus felismerésének témakörét. Ha lenne egy ilyen automatikus felismerő, ami bármilyen szoftverbe beépíthető lenne, akkor az nagyban elősegítené az álhírek elrejtését az emberek elől, vagy éppen az ilyen híreket közlő források felismerésében is segíthetne.

A mély tanulás, és azon belül is a mély neurális hálózatok manapság egyre népszerűbbek a tudomány területén, mert a legtöbb feladatban pontosabbak más gépi tanulási algoritmusoknál. Épp ezért úgy gondoltam, hogy a dolgozatomban a mély tanulás segítségével betanítok egy mély neurális hálózatot, a magyar nyelvű álhírek felismerésére. Tanítási módszernek a felügyelt tanulást választottam, ezért szükségem volt arra, hogy kiépítsek egy adatbázist, ami tartalmaz magyarul írt híreket, és álhíreket egyaránt.

Dolgozatom arra koncentrál, hogy hogyan jutottam el egy relatív pontos (97%-os pontosságú) felismerési képességgel rendelkező mély neurális hálózati modell kifejlesztéséhez. A dolgozat végén szót ejtek arról is, hogy hogyan lehetne ezt a modellt használni arra, hogy akár a mindennapi életünket is álhírmentessé tehesse.

2. Mély tanulás alapjai



2.1. ábra: A mély tanulás elhelyezkedése a mesterséges intelligencia témakörében. [leet.hu]

Ebben a fejezetben röviden ismertetem a mély tanulás és a neurális hálózatok alapjait, közel sem törekedve a teljességre, hiszen ez a témakör hatalmas, és külön könyveket lehetne róla írni. A fejezet célja csupán az, hogy alapismeretet adjon a kutatással kapcsolatos témakörökbe. A dolgozat végén található az ajánlott irodalom szekció (13. fejezet), amely alatt megtalálhatóak a témakör szélesebb ismereteit bemutató könyvek [20].

2.1. Matematikai alapok

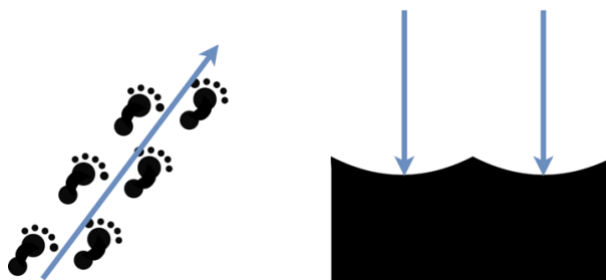
Ha a lentről felfele haladó megközelítéssel közelítünk a téma felé, akkor azt láthatjuk, hogy a mély tanulás és az alapjait adó neurális hálózatok működésének megértése érdekében elengedhetetlenek egyes matematikai ismeretek. Ezek többnyire lineáris algebrai fogalmak, amelyeket az alábbiakban röviden ismertetek [6].

2.1.1. Skalár

Az egyik legegyszerűbb fogalom a lineáris algebrában a skalár, hiszen skalár alatt általában egy számszerű értéket értünk. Olyan dolgokat fejezünk ki vele, amelyeknek nem lehet iránya. Gondolhatunk itt fizikai példákra, például: hőmérséklet, magasság, súly stb.

2.1.2. Vektor

Bizonyos adatok számszerűsítéséhez szükséges egy irányt is megadnunk. Ez a vektor, egy iránnyal rendelkező matematikai fogalom. Tulajdonképpen egy vektor egy csoport skalár, amely egy közös iránnyal rendelkezik. Itt is célszerű fizikai példákra gondolnunk, mint a sebesség, az erő, az elmozdulás stb. Ezek mind olyan események amikor szükségünk van egy értékre és egy irányra.



2.2. ábra: bal oldalon az elmozdulásvektor (lábnyomokkal), jobb oldalon a rugalmas testen alakváltozást okozó erő vektora

A vektorokat általában kisbetűvel írjuk le egy nyíllal a név felett, a könnyű megkülönböztethetőség érdekében. Lentebb látható egy n elemű, $\vec{a}$ névvel ellátott vektor matematikai leírása.

$$\vec{a} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad (2.1.)$$

Ahol a vektor elemei az x -el jelölt elemek, és összesen n skalár található benne.

2.1.3. Mátrix

A mátrixok, hasonlóképpen a vektorokhoz, az előző struktúrán alapulnak. Tulajdonképpen, ha veszünk több vektort és csoportosítjuk őket az elemeik száma alapján, akkor egy mátrixot kapunk. A mátrixokat általában vastagon szedett nagy betűkkel különböztetjük meg. Vegyünk példának három egyforma méretű vektort.

$$\vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}, \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}, \vec{c} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} \quad (2.2.)$$

Ha ezeket a vektorokat úgymond egymás mellé helyezzük, akkor egy táblázatnyi számot kapunk.

$$\mathbf{A} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & c_4 \end{bmatrix} \quad (2.3.)$$

Ezt a struktúrát nevezzük mátrixnak.

2.2. Alapvető fogalmak

Az alábbiakban röviden ismertetem azokat a fogalmakat [7, 8], amiket fontosnak tartok megismerni a kutatás mélyebb megértése érdekében. A kutatás során egy mesterséges intelligencia modellt használtam, azon belül is egy olyan gépi tanulási módszert, amit leggyakrabban mély tanulásnak neveznek. A 2.1. ábrán látható sorrendben mutatom be a fogalmakat (jobbról balra).

2.2.1. Mesterséges intelligencia fogalma

A mesterséges intelligencia (röviden: MI) fogalma és még az intelligencia fogalma is önmagában egy sok vitát tápláló kérdéskör lehet. Többféle megközelítése lehetséges a fogalom meghatározásának, így összesen négy fogalmat ismertetek az alábbiakban [7]:

- a) Egy tanulmányi terület a számítástechnika területén. A mesterséges intelligencia olyan számítógépek fejlesztésével foglalkozik, amelyek képesek részt venni olyan emberszerű gondolkodási folyamatokban, mint a tanulás, az érvelés és az önkorrekció.
- b) Az az elképzelés, hogy a gépek továbbfejleszthetők, hogy felvegyenek bizonyos képességeket, amelyeket általában az emberi intelligenciából valónak gondolunk, mint például a tanulás, az alkalmazkodás, az önkorrekció stb.
- c) Az emberi intelligencia kiterjesztése a számítógépek használatával, ahogyan a múltban a fizikai erőkiterjesztése mechanikus eszközök használatával történt.
- d) Szűk értelemben a számítógépek hatékonyabb használatának technikáinak tanulmányozása a továbbfejlesztett programozási technikák révén.

Mivel a dolgozatban ismertetett kutatás egy emberi tulajdonsággal (az álhírek felismerése) és azzal foglalkozik, hogy hogyan lehet rávenni egy gépet, hogy eltanulja ezt a tulajdonságot, ezért jelen helyzetben a b) fogalmat tekintem a legpontosabbnak.

2.2.2. Gépi tanulás fogalma

A gépi tanulás fogalmára a legelterjedtebb fogalom Tom Mitchell-től [8] származik, ami magyarul így hangzik:

- Olyan számítógépes programot nevezünk gépi tanulási programnak, ami *tapasztalatból* (E) tanul, amit egy bizonyos *feladat* (T) végzésével szerez, aminek végzését egy bizonyos *teljesítménymérték* (P) méri és ha teljesítményét (P) a feladat (T) végzése során megszerzett tapasztalattal (E) javítja.

Alapjában véve gyakorlatias szemmel elmondhatjuk, hogy egy gépi tanulási programhoz két dologra van szükségünk:

- Egy gép tanuló algoritmusra
- Egy adathalmazra

Az általam kiválasztott gépi tanulási modell a mély tanulás lett, aminek fogalmát az alább ismertetem.

2.2.3. Mély tanulás fogalma

A mély tanulás jelenleg rengeteg területen a legsikeresebb gépi tanulási módszernek bizonyult, olyannyira, hogy bizonyos területeken jobban teljesít, mint az ugyan azt a feladatot elvégző emberek [9, 10]. A mély tanulás fogalma:

- A mély tanulás olyan algoritmusok összessége a gépi tanulásban, amelyek több szinten próbálnak tanulni, az absztrakció különböző szintjei szerint. Általában mesterséges neurális hálózatokat használnak. Ezekben a tanult statisztikai modellekben a szintek a fogalmak elkülönült szintjeinek felelnek meg, ahol a magasabb szintű fogalmakat alacsonyabb szintű fogalmakból határozzák meg, és ugyanazok az alacsonyabb szintű fogalmak segíthetnek sok magasabb szintű fogalom meghatározásában [11].

Gyakorlatiasan kifejezve: a mély tanulás olyan gépi tanulási algoritmus, ami neurális hálózatokkal működik, de azon belül is mély neurális hálózatokkal, azaz olyan matematikai struktúrákkal, amik kapcsolati láncba fűzött mátrixokból állnak. Ezeket tanítjuk egy algoritmus segítségével, de végeredményben elmondhatjuk, hogy a különböző rétegek/szintek/mátrix elemek különböző absztrakciós szinteknek felelnek meg.

3. Szakirodalmi áttekintés

A dolgozatban bemutatott munka megkezdése előtt szakirodalmi áttekintést végeztem, hogy megbizonyosodhassak róla, hogy a tervezett munkának van-e tudományos értéke. Ebben a fejezetben bemutatom a fontosabb szakirodalmi írásokat, amik a dolgozatomhoz köthetőek és azok hatását a kutatásomra. Minden alfejezet egy ilyen szakirodalmi írásról szól, ezekben az alfejezetekben az aktuálisan vizsgált kutatásra csak úgy hivatkozom, mint „a kutatás”.

3.1. Vincze Veronika – Kacsa vagy nem kacsa? Magyar nyelvű álhírek automatikus azonosítása

Hasonlóképpen a dolgozatomhoz, a *Kacsa vagy nem kacsa? Magyar nyelvű álhírek automatikus azonosítása* című kutatás is a magyar álhírek felismerésével foglalkozott.

Az „álhírek” címszó alatt a kutatás a szórakoztatási célból megírt híreket érti, ez különbözik a kutatásom alatt értett álhír definícióval [13]. A kutatásban említésre kerül, hogy az angol nyelven írt álhíreket többnyire szintaktikai vagy nyelvi jellemzők alapján lehet a legkönnyebben megkülönböztetni, így ezeket a módszereket alkalmazza a magyar nyelven is. A kutatás céljára egy 80 hírből álló adathalmazt gyűjtöttek össze, amiben a valós híreket több országos és megyei hírportálról, míg az álhíreket az előbb említett weboldalak április elsejei cikkeiből, és egy parodizáló hírportál híreiből gyűjtötték össze.

3.1. táblázat

Szövegek	Pontosság
Hírek	78,6%
Álhírek	81,6%
Összesen	80,1%

A begyűjtött adatok elemzése után az álhírek felismerésére egy döntési fa modellt választottak, amely 80%-os pontossággal volt képes beazonosítani a híreket (azaz 80 mintából 64-et helyesen osztályozott), ez merőben eltér mind a kutatásom alatt használt modelltől, mind az elért pontosságtól. A tesztek eredményei megtekinthetők a 3.1. táblázatban.

Összességében a kutatás célja azonos a dolgozatomban szereplő kutatásával (álhírek felismerése mesterséges intelligenciával), de inkább egy váz lehet a kutatásom számára, mint kiinduló pont. Hiszen mind az álhír fogalmi eltérése, mind az adatok forrásai és a kiválasztott mesterséges intelligencia modell is eltér az általam választottaktól.

3.2. Veszelszki Ágnes - Az álhírek extra- és intralingvális jellemzői

Fontos alapot adott munkámhoz kiindulási pontként az álhírek pontos definíciója: „A megnevezés olyan hírcikkekre vonatkozik, amelyeknek nincs tényszerű alapjuk, azaz hamisak, de hírként jelenítik meg őket.” [13]. Ezért a témában remek összefoglalót adott *Az álhírek extra- és intralingvális jellemzői* című kutatás.

A cikk megemlíti több fontos információt is az álhírekről. A bevezetőben már említettem, hogy az álhírek többnyire a közösségi oldalak, és az Internetelés egyre széleskörűbb terjedésének segítségével nyernek, egyre nagyobb teret a mindennapokban. Ezen felül a kutatás megemlíti, hogy különböző módokon lehet kategorizálni az álhíreket:

- Terjedésük szándékossága szerint
- Terjesztésük módja szerint

A kutatásom szempontjából a terjedés szándékossága nem jelent fontos szempontot. Ezért inkább a terjedési mód lehet fontos, azon belül is az 1. „Eredeti híroldalakat utánzó, a márkájuk hitelességét kiaknázó, hamis oldalak” és a 2. „Álhíroldalak”. A kutatás végén található egy lista, úgynevezett „feketelista”, amely 126 különböző álhír forrást tartalmaz, a kutatásom során ezekről a weboldalakról történt az adatgyűjtés.

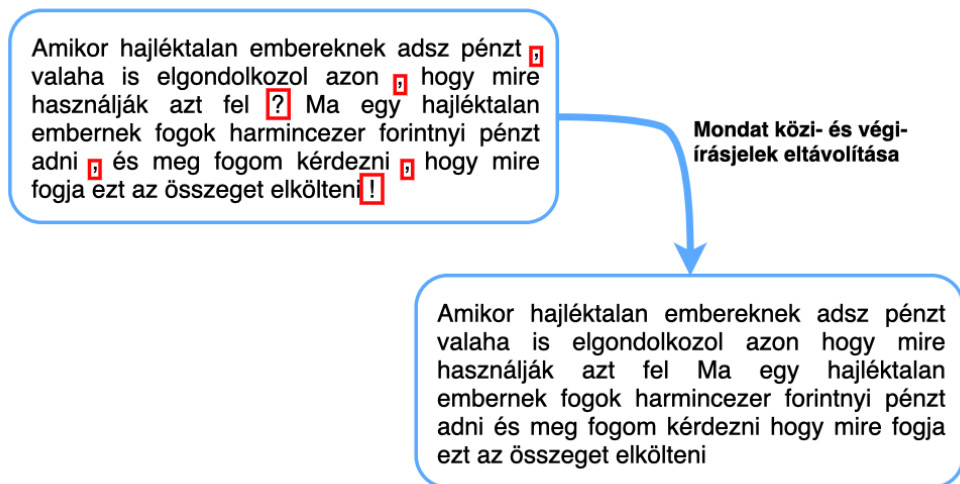
Összeségében elmondható, hogy a kutatás fontos alapot ad a dolgozatomhoz. Az említett definíciók megalapozzák a kutatásomat, a feketelista pedig remek kiinduló pontot ad az álhírek gyűjtéséhez.

3.3. Thota A., Tilak P., Ahluwalia S., Lohia N. - Fake News Detection: A Deep Learning Approach

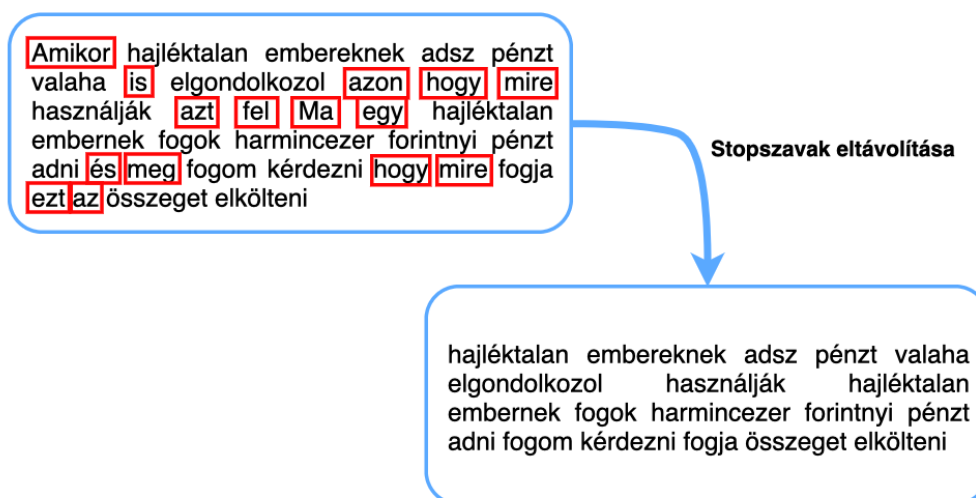
Természetesen utána jártam annak, hogy a témában már végeztek-e kutatást az említett fogalom alapján vett álhírek azonosítását illetően. A fejezetcímbe említett kutatás [19] mély neurális hálózatokat használ ezen cél érdekében, ami ugyancsak megfeleltethető a kutatásomnak, ám az eltérés az, hogy az angol nyelvel való munka teljesen más algoritmusokat igényel, mint a magyar nyelvvél való feladat. Megemlítendő még, hogy az én kutatásomtól eltérően nemcsak a hírek címeit, hanem a hír szövegét is felhasználták a beazonosításhoz.

Ennek ellenére a kutatásban említett mély tanulási modellek egy kiinduló pontot adtak a saját modelljeim struktúrájának megtervezéséhez. Illetve, a kutatás segített megállapítani, hogy miként kell az angol nyelvi adatokat értelmezhetővé tenni egy mély neurális hálózatnak, ezzel rávezetve engem a magyar nyelvi feldolgozásra ugyan ezen célból.

Az angol nyelven megírt híreket többféle módszerrel is feldolgozták a további munkákhoz. Mondat szinten minden mondat közeli- és végjel el lett távolítva, amik nem is lettek helyettesítve semmivel. Szavak szintjén minden stopszó (stopword) el lett távolítva és csak a szótörzseket hagyták meg. Az utóbbi módszert nevezzük szótövezésnek (stemming) [24]. A feldolgozási folyamatra láthatóak példák a 3.1. és 3.2. ábrán.



3.1. ábra: A mondat szintű feldolgozás szemléltetése példa mondattal



3.2. ábra: Példa a szavak szintjén történő feldolgozásra, a 3.1 ábrán látható szöveget tovább használva
Ezt az alap szövegfeldolgozást használták különböző adatfeldolgozó technikák hatékonyságának mérésére úgy, hogy kiválasztottak egy mesterséges intelligencia modellt és annak pontosságát mérték egy technikával párosítva.

Ami a kutatásban felhasznált modelleket illeti, három féle mély neurális hálózati modell típust használtak fel:

- Teljesen kapcsolt rétegekkel ellátott (DNN: Dense Neural Network)
- Konvolúciós rétegekkel ellátott (CNN: Convolution Neural Network)

- Visszacsatolt rétegekkel ellátott (RNN: Recurrent Neural Network)

Ezekon hálózatokon tesztelték az alábbi hiperparamétereket (zárójelben az a paraméter, amit optimálisnak ítélt meg a kutatás):

- Aktivációs függvények: softmax, relu, tanh (**softmax**)
- Köteg méret (batch size): 32 és 256 között (**64**)
- Lemorzsolódási ráta (dropout rate): 0 és 1 között (**0,1**)
- Tanuló ciklusok száma (epoch): 50 és 200 között (**50**)
- L2 büntetés (L2 penalty): 0,1 vagy 0,01 vagy 0,001 vagy 0,0001 (**0,0001**)

Összességében megállapítható, hogy ez a kutatás egy alapot adott egy a magyar nyelvre optimalizált szövegfeldolgozási technika megállapításához, és az alap mély neurális hálózatok megtervezéséhez, amikből kiindulva és némi változtatásokat eszközölve eljutottam a kutatásom eredményét adó mély neurális hálózati struktúrához.

3.4. Godbole V., Dahl G. E., Gilmer J., Shallue C. J., Nado Z. - Deep Learning Tuning Playbook

A címben említett kutatás [72] a mély tanulási modellek finomításáról, és a hozzá kapcsolódó folyamatokról szól. A kutatás célja az volt, hogy egy amolyan gyűjteménye legyen a mély tanulási modellek finomításához használt technikáknak.

A fejlesztési folyamat sikerességének érdekében fontosnak tartottam, hogy megismerjem a mély tanulási modellek finomításának legfontosabb aspektusait és bevált technikáit. A kutatásom során, a modell fejlesztéséről szóló fejezetben felhasználtam az ebből a kutatásból tanultakat, hogy a dolgozatomban modellezzem a lehető legjobb eredményeket érje el az álhírek felismerése terén. Ilyen felhasznált technika például:

- **A kezdeti modell struktúrájának megállapítása:** Egy az angol nyelvre írt álhírfelismerő mély neurális hálózat struktúrája (3.3 fejezet) alapján indulok ki
- **A kezdeti hiperparaméterek megállapítása:** Figyelembe véve az adathalmaz tartalmát, a tanításhoz használt hírcímek számát, a modell méretét és típusát is
- **A modell módosításának módszerei:** Segített megérteni, hogy a különböző paraméterek módosítása milyen hatással van a modell teljesítményére

Összességében véve a kutatás segített a modell fejlesztésének minden lépésében. Többek között segített megérteni, hogy milyen módosításokat érdemes eszközölnöm, a modell teljesítményének javítása érdekében.

4. Felhasznált technológiák

Ebben a fejezetben röviden bemutatom a kutatás során felhasznált szoftver és hardver technológiákat. Törekedtem az ingyenesen elérhető, esetleg nyílt forráskódú eszközök használatára, ezzel megkönnyítve azt, hogy a későbbiekben mások is részt vehessenek a kutatás tovább fejlesztésében.

4.1. Szoftvereszközök

A legfőbb felhasznált szoftvereszköz maga a Python [14] programozási nyelv volt, a kutatás során ezzel a nyelvel írtam meg magát a mély neurális hálózatot és az adatok begyűjtésére, illetve szűrésére alkalmas programokat. A nyelv 3.8.13-as számú verzióját használtam. Választásom azért esett erre a nyelvre, mert a Python programozási nyelven érhető el a legtöbb olyan API könyvtár, amely a gépi tanulási algoritmusok tanítását és használatát mind egyszerűbbé teszik, ezzel kevésbé kellett a kutatás során a kódra koncentrálnom, mint inkább arra, hogy mit várok el a program működésétől.

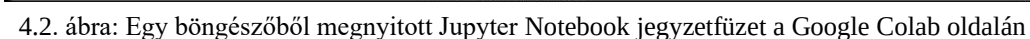
A programokhoz legjobban kapcsolódó szoftver, amelyet használatba vettem, az a Git (2.31.1) [15] verziókezelő rendszer volt. A kutatás során kifejlesztett szoftvereket egy repository-ban (kódtárban) tároltam, amelyet a GitHub [16] nevű oldalon keresztül használhattam. A szoftvereken felül az adatbázist is online tároltam [84].

A kutatás néhány részéhez tartozó fájlok és forrásfájlok elérhetőek a Hugging Face [17] nevű ingyenesen használható kódtároló oldalon. Erre az oldalra is a Git verziókezelő segítségével töltöttem fel a programkódokat. A dolgozat során készített és betanított mély neurális hálózati modellt, közvetlenül az oldalon keresztül kipróbálhatóvá is tettem, a Hugging Face eszközeinek segítségével. Így a kutatás végleges modelljét, illetve a szövegfeldolgozás módszerét bárki kipróbálhatja a böngészőjéből [82 vagy 76]. Ezzel a kutatást és eredményeit közérthetőbbé teszem.

Förtelmes lények másznak elő az olvadó gleccser mélyéről - VIDEÓ

[illegible]

fejlesztés során a Visual Studio Code (VS Code) névre hallgató, ingyenes, nyílt



..... (TYT) főállású tántorok: Fekete - főállású - VS Csak a helyi munkaadóknak van

Az sajátkezü adatgyűjtési folyamat során a Safari webböngészőt használtam, ám a scraper programok a szabad forrású Firefox böngészővel dolgoztak. A dolgozatban látható ábrákat pedig a diagrams.net [71] oldalon készítettem.

4.2. Szoftverkönyvtárak

A Python programozási nyelvre rengeteg szabad licenszű szoftverkönyvtár van, amelyek közül sok a mesterséges intelligencia témakörében hasznosítható a leginkább, ezek közül a könyvtárak közül az egyik legtöbbször használt a Tensorflow [21] 2.9.2-es és az arra épülő Keras [22] 2.9.0-ás verziója, amelyet a neurális hálózatok létrehozására és tanítására használtam a kutatás során. Egyéb a gépi tanulással kapcsolatos segédfüggvények használatára a Sklearn vagy Scikit_Learn [23] 1.1.2-es verzióját használtam.

A természetes nyelvi feldolgozásra használtam az igencsak nagy népszerűségnek örvendő NLTK könyvtár 3.7-es verzióját, illetve a Hugging Face által fejlesztett Transformers 4.22.2-öt. A Transformers könyvtáron keresztül jutottam hozzá a huBERT [4] modellhez, amit a kutatásom során néhány tesztelt neurális hálózat esetében fel is használtam.

Az adatgyűjtéshez legtöbbször általam megírt un. scraper programokat használtam, melyek a weboldalak kliens oldali HTML/CSS/JavaScript kódjában keresik a szükséges adatokat, ezen programok segítségével gyűjtöttem be a hírek/álhírek nagy százalékát. Ezekhez a programokhoz a Selenium 3.141.0-ás verziójú könyvtárat használtam.

Egyéb a diagrammok vagy függvények ábrázolására, illetve a szoftverfejlesztési munka elősegítésére használtam a Numpy 1.23.1, Matplotlib 3.6.0 és a TQDM 4.57.0-ás verziójú könyvtárakat.

4.3. Hardvereszközök

A kutatás során a kód megírására, az adatgyűjtésre és a modellek kipróbálására mind a saját laptopomat, egy Apple MacBook Air-t (2021) használtam. Ezen kívül felhasználtam a Hugging Face és a Google Colab által nyújtott, előfizetéssel elérhető felhőtárhelyeket és felhőn keresztül elérhető számítógépeket. A Hugging Face a már betanított modell bemutatásához volt szükséges (4.1. ábra), míg a Google Colab a nagy memória igényű tanítási folyamathoz volt használva. A kutatás során elérhető erőforrásaimat részletesen ismerteti a 4.1. táblázat.

4.1. táblázat

Erőforrás neve	RAM mérete	GPU RAM mérete
Apple MacBook Air	8 GB	8 GB
Google Colab	83,5 GB	40 GB
Hugging Face	16 GB	-

A neurális hálózatok számítási folyamataihoz a saját gépemen a CPU-t használtam, míg a Google Colab rendszerében a neurális hálózatokra optimalizált TPU-t [36] és a CPU-nál nagyobb teljesítményű GPU-t [37] vettem használatba.

5. Adatgyűjtés és feldolgozás

Ebben a fejezetben bemutatom milyen módon jutottam hozzá a felhasznált adatokhoz, azokból hogyan és milyen adatbázist építettem, illetve, hogy hogyan szűrtem és később hogyan dolgoztam fel az adatokat a mély neurális hálózat által „érthető” adatokká.

5.1. Adatok gyűjtése

Ebben a szakaszban részletezem, hogy milyen technikákat használtam az adatok begyűjtésére. Először is létrehoztam két fájlt, egy fájlt az álhírek és egy fájlt a hírek tárolására, mind a két fájl egyszerű szöveg formátumú, ahol az adatokat az új sor karakter (\n) választja el.

Adatok alatt a hírek címeit értem, ezeket egyszerű szöveg formájában szerzem be különböző weboldalakról, amelyek újságírással és hírek kiadásával foglalkoznak. A forrás weboldalak nyilvántartására összeállítottam két listát, az egyik listában az általában megbízhatónak nevezhető weboldalak címeit helyeztem el, a másikban pedig az úgynevezett feketelistában [13] található webcímek foglalnak helyet. A megbízható weboldalak listája tulajdonképpen olyan híroldalakkból állt, amik nem találhatók meg a feketelistában, de rajta vannak az *„Internetes közönségmérési adatok (2022 I. negyedév)”* [80] című kutatás, *„A tíz legtöbb interakciót elérő oldal”* listáján. Céлом volt, hogy a szöveg formában fellelhető hírcímeket a kettő közül az egyik fájlba gyűjtssem, amiben minden hírcímet egy külön sorba helyezek el, az egyszerű és egyértelmű elkülönítés érdekében. Természetesen odafigyeltem arra, hogy minden hírcím csak egyszer szerepeljen az adatbázisban, és hogy csak egy kategóriában legyen fellelhető.

Az adatbázis tehát binárisan osztályozott adatokat tárol. A továbbiakban azokra a hírekre, amik nem álhírek, „valós hír” vagy egyszerűen „hír” néven hivatkozok.

Bár ebben a részben még csak az adatok gyűjtése a feladat, itt is szükséges egy amolyan előszűrést bevezetni, melynek során odafigyeltem arra, hogy minden hírcím megfeleljen az alábbi kritériumoknak:

1. Tartalmazzon legalább három szót
2. Magyar nyelven íródjon
3. Egyértelműen meg lehessen határozni, hogy álhír vagy hír

A felsorolt feltételekből az elsőt egy egyszerű algoritmus is képes ellenőrizni. A második felvételt is egyszerűbb ellenőrizni a weboldal előzetes átvizsgálásával. Azonban a harmadik feltétel az, amikor már külön szűrésre lenne szükségem. Ezt a szűrést úgy oldottam meg, hogy

ha egy hírcím forrása egy olyan weboldal, ami a megbízható oldalak listájában szerepel, akkor hírként csoportosítom, ellenkező esetben az álhírek sorát gyarapítja.

5.1.1. Sajátkezű gyűjtés

Sajátkezű gyűjtésnek nevezem, amikor én egy böngésző használatával az Interneten fellelhető különböző hírportálokat böngésem, és ha egy adott hírcímet látok, akkor azt lemásolom az adatokat gyűjtő fájlok egyikébe. Ehhez a feladathoz a gépemen előre telepített Safari böngészőt használtam. Körülbelül 50 hírt és 50 álhírt gyűjtöttem ezzel a technikával.

5.1.2. Automatizált gyűjtés

Itt az általam írt programok használatával való gyűjtésre gondolok. Ez az a technika, amikkel az adatok körülbelül 95%-át (3205 hírcím) gyűjtöttem be különböző forrásokból. Erre a feladatra scraper programokat [49] használtam, amiket természetesen én magam írtam meg. A programok a Firefox webböngészőt használták távvezérlés módban, és az előre meghatározott webcímek listájából szedték a forrás webcímeit. Így a programok szempontjából egyszerű volt a kategorizálás, ám mivel az egész folyamat automatikus volt, ezért a begyűjtött címeket szükséges volt utó ellenőrizni, hogy a megfelelő kategóriába kerültek-e.

Itt még fontos azt is hozzátennem, hogy nem volt elég egy scraper program megírása, hiszen minden weboldal HTML kódja más, így tulajdonképpen a listákban fellelhető minden webcímhez (50 darab) az adott weboldal szerkezetéhez szabott programot kellett írnom. Hiszen egyes weboldalakon lehet, hogy a hírcímek a H2-es HTML elemekben találhatóak, más weboldalakon lehet, hogy egyszerű P elemek, és csak egy bizonyos CSS osztállyal vannak elkülönítve a többi elemtől.



5.1. ábra: A Firefox webböngésző távvezérlés módban megnyitva a 24.hu oldalon [24.hu]

Mindezekhez olyan Python programokat írtam, amik a Selenium könyvtár használatával távvezérlés módban nyitják meg a Firefox webböngészőt. A folyamat során a program megnyit egy adott webcímet, ez után pedig az általam meghatározott szabályok alapján elkezd keresni a hírcímeket az éppen megnyitott weboldal HTML kódjában. Ha először töltöttem le hírcímeket az adott weboldalról, akkor első dolgom volt megnyitni az oldalt egy külön böngésző ablakban és az „Oldal forrásának megtekintése” módban meg kellett keresnem, hogy a hírcímeket hogyan

különböztethetem meg a többi HTML elemtől, ez többnyire az elemek CSS osztályaival megvalósíthatónak bizonyult.

5.1.3. Kérdőíves gyűjtés

Ez az a gyűjtési folyamat melynek során egy Google Űrlapok-on megjelenő és publikált kérdőív segítségével gyűjtöttem be néhány hírcímet. Az űrlap a *Magyar álhír felismerés – Álhír jelentő űrlap* [77] címet viseli és a következő szöveget tartalmazza:

Kedves Kitöltő,

Kondor Martin vagyok, a Pannon Egyetem Mérnökinformatikus hallgatója. Az alábbi kérdőív egy kutatásért jött létre, melynek célja egy olyan M.I. fejlesztése, ami automatikusan és megbízhatóan képes beazonosítani magyar nyelvű álhíreket.

Kérem segítségét abban, hogy ha Ön egy álhírral találkozik az Interneten, akkor ezen az űrlapon adja meg a hír címét, illetve webcímét, ezzel elősegítve a kutatásomat! A kérdőívet többször is használhatja újabb és újabb álhírek jelentésére.

Definícióm szerint: Az álhírek hírként bemutatott hamis vagy félrevezető információk.

Köszönöm, hogy kitöltésével segíti a munkámat!

Mindössze két bemeneti szövegmezőt helyeztem el az űrlapon, az egyik címkéje „Az álhír címe:”, a másik címkéje pedig „Az álhírré mutató webcím:”. A bemeneti mezők mellé írtam egy példát kitalált álhírré „*ÁTVERTÉK AZ EMBERISÉGET! A PIROS SZÍN VALÓJÁBAN KÉK!*”, és webcímre is „*http://example.com*”.

Az űrlapot 2022. szeptember 28.-án publikáltam és a kutatás írásának időpontjáig összesen 20 álhírt sikerült vele begyűjteni. Ezeket a hírcímeket természetesen ugyan úgy szűrni kell, hogy helyes kategóriában legyenek, de alapvetően ezzel az űrlappal csak egy kevés álhír gyűjtését tűztem ki célul.

5.1.4. Összefoglalás

Az 5.1. táblázatban szemléltetem, hogy a különböző adatgyűjtési módszerekkel pontosan hány hírcímet sikerült összegyűjteném a kutatás során.

5.1. táblázat

Gyűjtési módszer	Begyűjtött hírcímek száma	Begyűjtött álhírcímek száma
Kérdőíves	0	20
Automatizált	1631	1574
Sajátkezű	50	50
Összesen	1681	1644

5.2. Adatok szűrése

Ebben az alfejezetben bemutatom, hogy hogyan és milyen technikákkal szűrtem az adatokat. Szűrés alatt a felesleges, de már begyűjtött adatok törlését értem. A szűrésre azért van szükség, mert lehetséges, hogy az adathalmazba belekerülnek olyan hírcímek, amik esetleg helytelenül formázottak lehetnek vagy olyan nyelvtani hibákkal rendelkeznek, amely miatt nem tekintem őket értelmezhetőnek. Ezen felül fontos tény, hogy az adatok szűrése a tanulási fázis megkezdése előtt javítja a gépi tanulási modellek tanulási hatékonyságát [12].

5.2.1. Szoftveres szűrés

Mivel az álhírek a valós hírektől szókészletben mutatják a legnagyobb eltérést [2] így a szűrés akkor a leghatásosabb szoftveresen, ha a szavak alapján szűr. A szűrési folyamatra egy algoritmust írtam, amely bemenetként egy szöveg formátumú mondatot (hírcímet) fogad, amiről eldönti, hogy eldobandó (**VISSZATÉRÉS 1**) vagy felhasználható (**VISSZATÉRÉS 0**). Pszeudokóddal leírva a következőképpen működik:

5.1. algoritmus

```

1  FÜGGVÉNY ELDOBANDÓ(MONDAT)
2      L := SZAVAK_SZÁMA(MONDAT)
3      HA L <= 3 AKKOR
4          VISSZATÉRÉS 1
5      ELÁGAZÁS VÉGE
6      A := (STOPSZAVAK_SZÁMA(MONDAT) / LEN)
7      HA A >= 0.5 AKKOR
8          VISSZATÉRÉS 1
9      ELÁGAZÁS VÉGE
10     VISSZATÉRÉS 0
11  FÜGGVÉNY VÉGE

```

A kódban látható két függvény is, ami magyarázatra szorulhat:

- **SZAVAK_SZAMA:** a bemeneti mondatban megszámolja a szavak számát és ezt adja visszatérési értéként.
- **STOPSZAVAK_SZAMA:** visszatérési értéként adja, hogy a bemeneti mondatban hány stopszó (stopword) található.

Stopszó alatt a leggyakrabban használt 200 szót értjük a magyar nyelvben (például: „a”, „az”, „és”, „vagy” stb.), ez egy listányi szó, ami az NLTK programkönyvtár segítségével könnyen elérhető.

Az algoritmus tehát úgy van tervezve, hogy a rövid mondatokat (a kevesebb mint három szó hosszúakat), és azokat a mondatokat, ahol a stopszavak több mint felét kiteszik az összes szónak, kiszűri. Ezzel a módszerrel ellenőriztem mind a két halmaz teljes tartalmát.

5.2.2. Kézi szűrés

Kézi szűrés alatt azt a folyamatot értem, amikor a már meglévő adathalmazban a hírcímeket egyesével átolvasom és ha nem megfelelőnek találom, akkor kitörlöm a mondatot az adatbázisból.

A legegyszerűbb eset ebben a fázisban, hogy ha egy hírcím nem magyarul volt írva, akkor azt azonnal töröltem az adatbázisból. Máskülönb a legfőbb szempontom a lehető legpontosabban megállapítani, hogy egy hírcím valóban a megfelelő kategóriában van-e. Ehhez szükséges volt felhasználnom az álhírek definícióját [13], és ha egy hírcím nem felelt meg az alább felsoroltak egyikének, akkor töröltem:

- **kattintásvadász:** [50] figyelemfelkeltő vagy ritkán használt szavak jelenléte a címben (például: világrend, mikrochip, elgondolkodtató, mutatjuk, videó stb.).
- **összeesküvés-elmélet:** összeesküvés elméletek [51] említése a címben.
- **áltudományos:** áltudományos témák [52] említése a címben.

Ezzel a módszerrel néztem át a teljes álhír halmazt. A valós hírek forrásaira való tekintettel nem találtam szükségesnek, hogy azokat is ilyen alaposan kelljen átellenőriznem.

5.2.3. Szakértői szűrés

Ahhoz, hogy megbizonyosodhassak róla, hogy az álhírnek osztályozott adatok valóban hamis hírek-e, konzulensem segítségével felkértünk különböző szervezeteket és szakértőket, hogy átnézzék és hitelesítsék az álhírek adathalmazát. A dolgozat készítése során a hitelesítés még folyamatban volt.

5.3. Adathalmaz elemzés

Mielőtt elkezdtem a feldolgozást szükségesnek ítéltam, hogy a már szűrt adathalmazt egy pár szempontból elemezzem és megvizsgáljam. Ezzel megbizonyosodva róla, hogy az adathalmaz készen áll a feldolgozásra, illetve, hogy a két hírkategória tényleg megkülönböztethető-e



Szótérképeken kívül egy egyszerűbb gyakoriság számolást is végeztem, ami az 5.5. ábrán látható.



Az 5.2 táblázatban egyéb statisztikák találhatók az adathalmaztól.

5.2. táblázat

\	Adathalmaz mérete	Leghosszabb mondat	Mondatok átlag hossza	Leggyakoribb szó	Leggyakoribb mondat végi írásjel (a pont kivételével)
Hírek	1634 hírcím	161 betű / 24 szó	68 betű / 10 szó	magyar	?
Álhírek	1501 hírcím	168 betű / 31 szó	80 betű / 12 szó	videó	!

Érdekes, hogy a felmérés szerint az álhírek átlagban hosszabbak a híreknél, és hogy a leggyakoribb mondat végi írásjel rendszerint a kijelentő mondatokat jelöli, míg a hírekben található a kérdőeket.

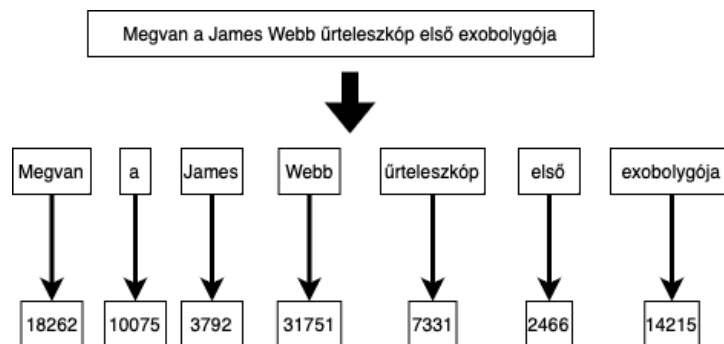
5.3. táblázat

Írásjel	Gyakoriság az álhírekben	Írásjel	Gyakoriság a hírekben
,	826	,	880
!	536	-	342
-	441	.	85
?	437	?	73
.	109	!	27

Ahogy látható az 5.3. táblázatból is, az írásjelek használatában nagy különbségek vannak a hírek és álhírek között.

5.4. Adatfeldolgozás

A neurális hálózatok nem képesek szövegekkel dolgozni, ezért szükség van egy módszerre, amivel a hírcímeket számszerűvé alakítom. Mivel abból a feltételezésből indulok ki, hogy az álhíreket meg lehet különböztetni a hírektől a szóhasználatuk vagy a használt írásjelek alapján, így a hírcímeket szavakra bontás (az írásjeleket külön szavaknak veszem) után alakítom számszerű adatokká. A mondatokat tehát szavakra bontom, és a szavakat számokká alakítom, így egy mondatból létrejön egy vektor, ahol egy adott i pozícióban lévő szám, mindig megfelel a mondatban található i . szóval. Erre a számszerűsítésre egy példa látható az 5.6. ábrán.



5.6. ábra: Szemlélteti, hogy szavakra bontás után, minden szóhoz egy számot egy rendelünk (adatfeldolgozás). Ezt a folyamatot nevezem adatfeldolgozásnak (tokenization), melynek során minden hírcím-ből egy pozitív egész számokból álló vektor lesz, ahol minden szám a hírcím-ben elhelyezett szónak felel meg.

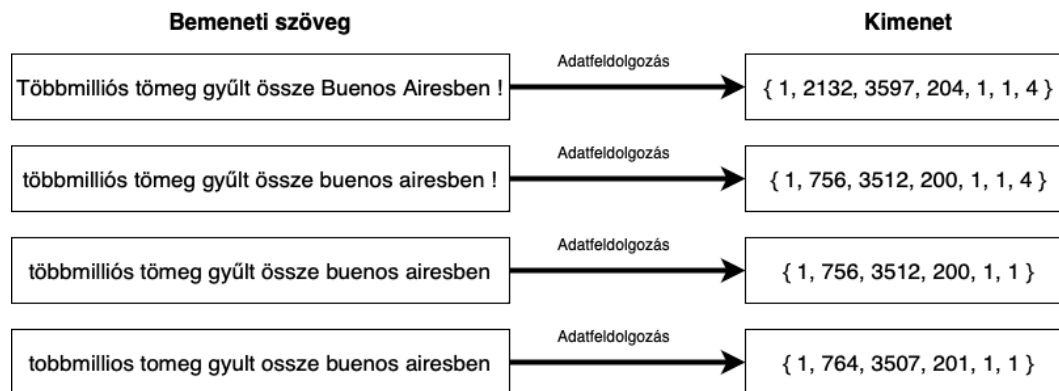
Mivel a vektorra alakítás során egy mondat minden szava egy számhoz kell, hogy rendelődön, így elsősorban a szöveg formai tulajdonságainak (ékezetek, kisbetű/nagybetű) megváltoztatásával kellett foglalkoznom, ezt nevezem szövegfeldolgozásnak. Hiszen, ha minden szó egy szám, akkor mondjuk a „Felszólít”, „felszólít” és „felszolít” szavak különböző számokhoz rendelődnek, miközben nem minden esetben jelentenek mást.

5.4.1. Adatfeldolgozási folyamat

Ebben a fejezetben röviden ismertetem, hogy milyen lépéseket alkalmaztam, a teljes adathalmaz feldolgozása érdekében.

5.4.1.1. Első lépés: Szövegfeldolgozás

Első lépésként alkalmazok egy szövegfeldolgozási módszert, amit az 5.4.2. fejezetben részletesebben is ismertetek. Erre azért van szükség, hogy a szövegfeldolgozott adathalmazból legyen előállítva a szókincstáblázat (egy olyan táblázat, ami minden szóhoz egy egyedi számot rendel, 5.4.1.2 fejezet, 5.8. ábra). Ha nem így lenne, és először hoznám létre a szókincstáblázatot, akkor megeshetne az a lehetőség, hogy egy szó átalakul a szövegfeldolgozás miatt, és később nem lesz megtalálható a szókincstáblázatban. Például, lehetséges lenne az, hogy az „AUTÓ” szó szerepel az adathalmazban, és tegyük fel, hogy a szövegfeldolgozás mindent kisbetűsre állít, ez esetben a szókincstáblázatban az eredeti szó szerepelne, de az adatfeldolgozó algoritmus miatt az adathalmazban csak az „autó” szó lenne megtalálható, amit aztán a szoftver nem találna meg a szókincstáblázatban. Különböző szövegfeldolgozási módszerek eredményeit mutatja az 5.7. ábra.



5.7. ábra: Az alapnak vett „Többmilliós tömeg gyűlt össze Buenos Airesben!” mondatra alkalmaztam többfajta szövegfeldolgozási módszert, aminek az eredményei láthatóak a bal oldalon, az ábrán látható, hogy nem mindegy, hogy egy adott szöveget milyen módon dolgozok fel, hiszen teljesen más számok jöhetnek ki kimenetként az adatfeldolgozó algoritmusból (amik az ábrán a jobb oldalon vannak szemléltetve)

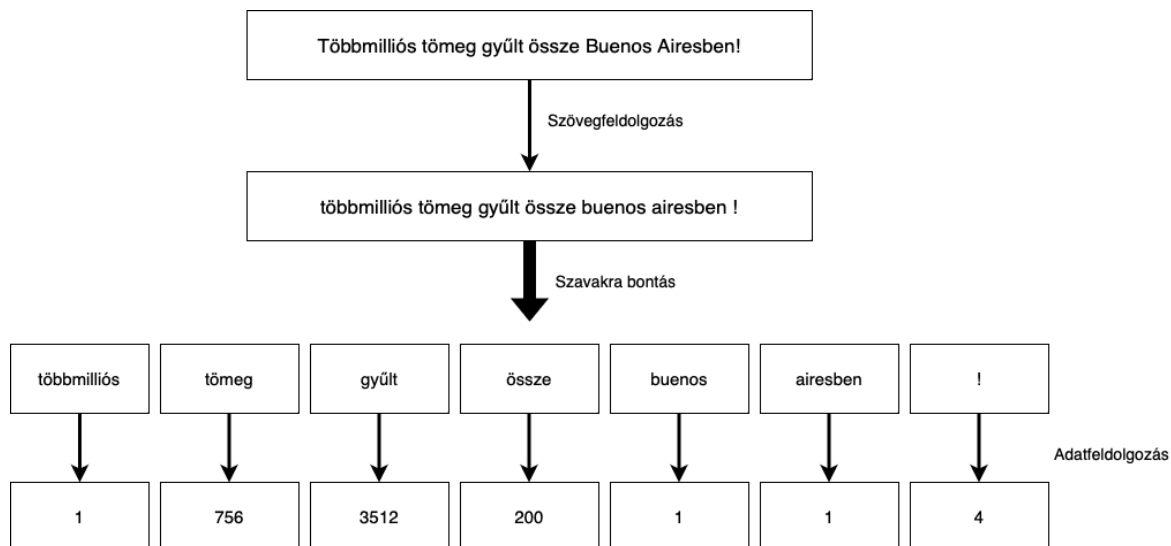
5.4.1.2. Második lépés: Szókincstáblázat

Minden szóhoz egy egyedi számot kell rendelni, hogy a neurális hálózat különbséget tudjon köztük tenni. Ehhez szükség van egy úgynevezett szókincstáblázatra, ami tulajdonképpen egy JSON objektum, ahol a kulcs maga a szó, és az érték egy a szóhoz rendelt szám. A táblázat létrehozására az Keras programkönyvtár Tokenizer osztályát használtam. Egy kisebb szókincstáblázatra látható példa az 5.8. ábrán.

```
{
  "<NULL>": 1,
  "a": 2,
  "az": 3,
  "!": 4,
  "tömeg": 756,
  "gyűlt": 3512,
  "össze": 200,
  .
  .
  .
}
```

5.8. ábra: Példa egy szókincstáblázatra, ha egy szót nem találunk a táblázatban, akkor a <NULL> szó számát fogja kapni, a „...” a JSON objektum végén jelöli, hogy a táblázat még más szavakat is tartalmaz az ábrán láthatóak mellett

Később, az ebben a lépésben létrehozott szókincstáblázat segítségével alakítok át minden szót egy számmá (példa az 5.9. ábrán).



5.9. ábra: Mint látható a „többmilliós”, „buenos” és az „airesben” szavak nem voltak benne a szókincstáblázatban, így a <NULL> értékét kapták, ami 1

5.4.1.3. Harmadik lépés: Számszerűsítés

Az adatfeldolgozás utolsó lépése, melynek során minden hírcímet a szövegfeldolgozás után, a szókincstáblázat segítségével vektorokká alakítok. Mivel a létrejött vektorok különböző hosszúságúak, ezért ebben a lépésben kiegészíték minden vektort, egy előre meghatározott hosszra. A kiegészítéshez használt szám a szókincstáblázatban található „<NULL>” szónak felel meg (5.8. ábra). Ha egy vektor hosszabb, mint a meghatározott hossz $K \in \mathbb{N}$, akkor a vektor méretét K -ra állítom, törölve a meghatározott hosszon túl eső számokat (amik előzőleg szavak voltak) a vektorból.

5.4.1.4. Összefoglalás

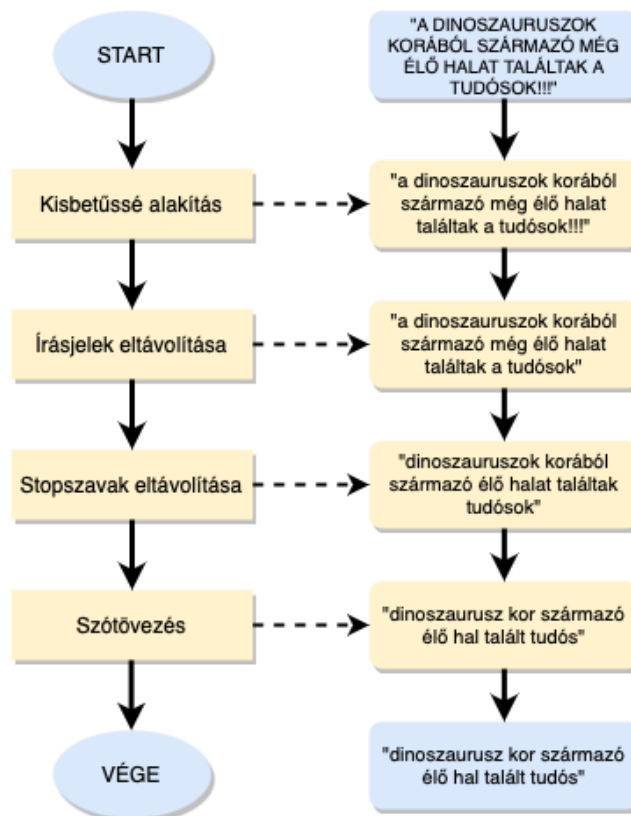
Az adatfeldolgozás célja csupán az, hogy a mély neurális hálózatok számára értehetővé tegye a mondatokat, azaz számszerűsítse. Az alább (5.4.2. fejezetben) ismertetett szövegfeldolgozó módszereknek arra kell törekedniük, hogy minél kisebb legyen a szókincstáblázat és kerülje a redundanciát, különben rengeteg felesleges szó keletkezik, amiket a neurális hálózat bár másképpen értelmez (pl.: „Kéz” vagy „kéz”) de valójában ugyan azt jelentik, tehát lehetne ugyan az a hozzájuk rendelt szám. Egy jól megválasztott szövegfeldolgozó algoritmus szemben a semmilyen feldolgozással, csökkentené a memóriahasználatot és a neurális hálózatnak is kevesebb szót kellene megtanulnia.

5.4.2. Szövegfeldolgozás

Ebben a fejezetben ismertetem a kifejlesztett szövegfeldolgozási algoritmust, amit adatelemzési fejezetben ismertetett információk alapján fejleszték. Ezen kívül ismertetem a megfontolandó változtatásokat, és a módosítások hatását a szókincstáblázatra.

Szövegfeldolgozásnak azt a folyamatot értem, melynek során az adathalmazban található hírcímeket módosítom, egy előre meghatározott algoritmus alapján. Erre azért van szükség, mert az előző fejezetben bemutattam, hogy a szövegfeldolgozástól függ a szókincstáblázat mérete, illetve az is, hogy a modell milyen szavak alapján tanulja meg a különbségeket az álhírek és hírek között.

Az alapnak vett algoritmus [19] (folyamatábra az 5.10. ábrán) szótövezéssel [24], stopszavak eltávolításával, minden betű kisbetűre állításával, és az írásjelek eltávolításával dolgozza fel a hírcímeket. Ebben a fejezetben bemutatom, hogy ezen az alapnak vett szövegfeldolgozási módszeren milyen módosításokat eszközöltem és mi lett a végső szövegfeldolgozó módszere a kutatásomnak. A különböző módosításokat mind ellenőrzöm az által, hogy milyen hatással vannak a felismerési pontosságra. A pontosság mérésére készítettem egy alap mély neurális hálózatot, amit aztán az adott feldolgozási módszerrel feldolgozott adatokkal tanítok, és tesztek végzek a felismerési pontossága megállapítására. Mivel a mély neurális hálózaton nem eszközölök változtatást, csak a feldolgozási módszeren, így látható lesz, hogy mely módszer jobb a hírcímek feldolgozására a felismerési pontosság tekintetében.



5.10. ábra: Folyamatábra az alapnak vett algoritmus működéséről, mellette pedig egy példa hírcím feldolgozása látható lépésről lépésre

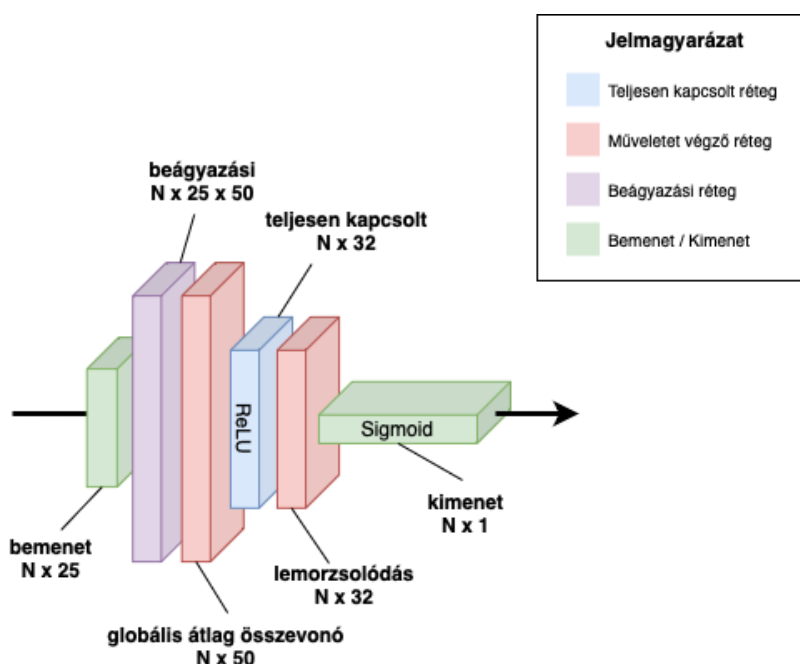
5.4.2.1. Teszt környezet ismertetése

Ebben a fejezetben ismertetem, hogy milyen tesztkörnyezetet állítottam fel az adatfeldolgozó algoritmuson végzett különböző változtatások ellenőrzésére. A fejlesztés sikerességének vizsgálatához egy teljesen kapcsolt rétegekkel ellátott neurális hálózatot vettem, aminek struktúráját az 5.11. ábrán szemléltetem.

Ez a neurális hálózat nem a végleges struktúra, csak referenciapontként van használva a fejezet erejéig. Az első réteg a bemenetet reprezentálja, ahol megmondja, hogy a már feldolgozott szövegből készült vektor maximum 25 szám hosszú lehet. A második egy beágyazási (embedding) [44] réteg. A középső egy globális átlag összevonó (global average pooling) [27] réteg. Ezek után egy teljesen kapcsolt réteg következik, aminek a kimenetein lemorzsolódást (dropout) [45] alkalmaztam. Végül az utolsó réteg, ami csupán egy kimenettel rendelkezik. A hálózat kimeneti rétege minden bemeneti hírcímhez rendel egy $\hat{y} \in [0; 1]$ valószínűségi értéket, amit úgy lehet értelmezni, mint egy valószínűségi értéket. A kimenet tehát annak a valószínűsége, hogy a bemenetként kapott hírcím, egy álhír (a modell tudása alapján).

Az adathalmaz véletlenszerű megkavarása után, 20% adatot félreraktam tesztelésre és 80% adatot a tanítási fázisra [28]. Ezután a tanító halmazon elvégzem az adatfeldolgozó módszert, és az ebből létrejövő adathalmazt használom a tanításra. A tanításhoz az Adam [25]

optimalizációs módszert alkalmaztam, 0,001-es tanítási rátával. Hibafüggvénynek pedig a Bináris Keresztentropiát [26] használtam (hiszen két osztályt kell azonosítani: álhír, hír). A tanítás során a tanító halmazt 64-es méretű kötegekre bontottam (minibatch) [30]. A tanító adathalmaz 20%-át validációs halmaznak [31] használtam fel. A teljes tanító ciklusok számát pedig lekorlátoztam 30-ra. Természetesen a tesztelés során megbizonyosodtam róla egy mag számmal (random seed), hogy a véletlenszerűen generált számok a neurális hálózat mátrixaiban minden teszt során ugyan azok legyenek, így a véletlenszerűség nem módosítja a mérési eredményeket.



5.11. ábra: Az ábrán látható a referenciaként használt teljesen kapcsolt rétegekkel ellátott neurális hálózat struktúrája, a réteg reprezentációján látható az aktivációs függvény neve, vonallal odakötve pedig a réteg típusa és a kimenetének dimenziói²

5.4.2.2. Előzetes mérések

Még mielőtt az alap módszeren módosításokat hajtottam volna végre. Végeztem egy előzetes mérést arról, hogy milyen pontossággal működik az alap módszer a tesztkörnyezetben. Az eredményeket és a következőkben használt mérési szempontokat az 5.4. táblázaton szemléltetem.

² Ezt az ábra stílust fogom használni mostantól a modellek reprezentálására, ezen az ábrán a rétegek különböző színű téglalapokként szerepelnek, hogy a rétegek típusai ránézésre megkülönböztethetők legyenek. A rétegek pontos nevét/típusát egy hozzájuk csatolt vonal jelzi. A rétegek neve alatt található, hogy a réteg által kimenetként adott mátrix milyen méretű (például: a bemeneti $N \times 25$ méretű mátrixot ad bemenetként a beágyazási rétegnek), ahol N a bemenetek száma (bemenetként kapott hírcímek). Az utolsó réteg (kimenet) fektetve van az átláthatóság kedvéért.

5.4. táblázat

Szempont	Eredmény
Szókincstáblázat mérete (szavak)	7859
Szókincstáblázat mérete (karakterek)	56 259
Szükséges tanító ciklusok száma	14
Pontosság (tanító halmazon)	~99,61%
Pontosság (teszt halmazon)	~79,51%

A szükséges tanító ciklusok száma azért lehet eltérő módosításonként, mert a neurális hálózatot egy úgynevezett korai leállás (early stopping) [32] módszerrel tanítottam, ami annyit tesz, hogy ha a pontosság a validációs halmazon nem nő már egy bizonyos számú tanítási ciklus óta (10-re volt beállítva), akkor leáll a tanítási fázis.

Mostantól, ha a szöveg külön nem tér ki rá a dolgozatban, akkor a pontosságot mindig a teszhalmaz módszerrel mérem le, amit a 7.1. fejezet részletesen ismertet.

5.4.2.3. Módosítás: Ékezetes betűk cseréje

Mivel az alapnak vett módszer angol nyelvre fejlesztett, így először azt próbáltam ki, hogy milyen hatása van az adatfeldolgozásnak, ha nincsenek ékezetek az adathalmazban. Tehát az ékezetes karaktereket helyettesítettem az ékezet nélküli megfelelőjükkel (például: „ó”-ból lett „o”, „á”-ból lett „a” stb.). Az eredményeket az 5.5. táblázat szemlélteti.

5.5. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
Szókincstáblázat mérete (szavak)	7829	-30
Szókincstáblázat mérete (karakterek)	56 646	+387
Szükséges tanító ciklusok száma	18	+4
Pontosság (tanító halmazon)	~99,51%	-0,1%
Pontosság (teszt halmazon)	~85,75%	+6,24%

Pozitív változás az, hogy a szókincstáblázat mérete a szavakat tekintve csökkent, így kevesebb szót kell megtanulnia a modellnek, ami látszik is a pontosságot nézve, hiszen a 6%-ot javult a teszhalmazon. Csökkent a pontosság a tanító halmazon, ám ez is pozitív változás, hiszen így látjuk, hogy a modell az alap módszerhez képest kevésbé hajlamos a túlillesztésre (overfitting) [33]. Negatívum csak az lehet, hogy tovább tartott a tanulás négy ciklussal, illetve, hogy a szókincstáblázat háromszáznolcvanhét karakterrel több memóriányi helyet foglal el.

5.4.2.4. Módosítás: Ékezetek és szótövezés nélkül

Tovább próbálva vinni az ékezetek nélküli verzió sikerességét leteszteltem, hogy mennyire teljesít jól egy olyan módszer, ahol az ékezetek elvétele által a szavak száma bár csökken (az

5.5. táblázatból kiindulva), de a szótövezés kihagyása miatt mégis nőnie kell a szókincstáblázat méretének. Az eredmények az 5.6. táblázaton láthatóak.

5.6. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
Szókincstáblázat mérete (szavak)	10 439	+2580
Szókincstáblázat mérete (karakterek)	86 978	+30 719
Szükséges tanító ciklusok száma	15	+1
Pontosság (tanító halmazon)	~99,90%	+0,29%
Pontosság (teszt halmazon)	~78,62%	-0,89%

Ez a módszer nem bizonyult jónak a kutatás szempontjából, hiszen a pontossága még az alap módszerhez képest is csökkent, illetve, a szótövezés kihagyása miatt jóval több egyedi szó került a szókincstáblázatba. Azért lett pontatlanabb, mert a tesztelt modell túl sok egyedi szót látott a tanulóhalmaz méretéhez képest, így nem tudta megtanulni, hogy milyen szavakat használnak jellemzően álhírekben vagy hírekben.

5.4.2.5. Módosítás: Stopszavakkal és ékezetek nélkül

A stopszavak eltávolítása rengeteg szót eltávolít a szókincstáblázatból, ami egy pozitívum, de ettől függetlenül lehetséges, hogy az álhíreket és híreket a stopszó használati mintázatuk is megkülönbözteti, így érdemesnek tartottam megpróbálni ezt a módszert. Az eredmények láthatóak az 5.7. táblázatban.

5.7. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
Szókincstáblázat mérete (szavak)	7899	+40
Szókincstáblázat mérete (karakterek)	57 065	+806
Szükséges tanító ciklusok száma	20	+6
Pontosság (tanító halmazon)	~99,95%	+0,34%
Pontosság (teszt halmazon)	~84,86%	+5,35%

A szókincstáblázat mérete nőtt, ám nem annyira, hogy az megzavarja a tanulási folyamatot. Ez meglátszik a pontosságban is, hiszen az eredmények azt mutatják, hogy ez a módszer javítja a pontosságot az alap módszerhez képest.

5.4.2.6. Módosítás: Kisbetűkre váltás és szótövezés nélkül

Látva, hogy az álhírek többsége nagybetűs szavakat használ, érdemesnek tartottam megpróbálni azt, hogy nem állítom kisbetűre a szavakat, hanem hagyom, hogy a teljes természetességükkel legyenek feldolgozva. A szótövezést pedig azért kapcsoltam ki, mert úgy

találtam, hogy látszólag túl sok tartalom vész el általa egy mondatból. Az eredményeket az 5.8. táblázat szemlélteti.

5.8. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
Szókincstáblázat mérete (szavak)	11 728	+3869
Szókincstáblázat mérete (karakterek)	96 404	+40 145
Szükséges tanító ciklusok száma	18	+4
Pontosság (tanító halmazon)	~100,0%	+0,39%
Pontosság (teszt halmazon)	~92,44%	+12,93%

Az eredmények pozitívnak tűnhetnek, hiszen a pontosság 12%-kal javult, ám a tanító halmazon mutatott tökéletes beazonosítási képesség jelzi a túlillesztés jelenlétét. Ezen felül meg kell említeni azt is, hogy attól, hogy a tanító halmazban található álhírek nagy része csupa nagybetűvel van írva, még nem általánosíthatunk arra, hogy minden álhír nagybetűvel lesz írva, amit a modell bemenetként fog kapni (ha egy álhírt kisbetűvel írnak, akkor a hálózatot az megtéveszti és valós hírnek fogja tekinteni, mert megtanulta, hogy az álhírek nagybetűvel íródnak). Így az eredmények bár kecsegtetőek de nem rendelkeznek valós értékkel a kutatás célját tekintve. Ezen okokból, a szövegek kisbetűsre állítása a szövegfeldolgozási folyamat része maradt, a kutatás végső modelljének tanítása során.

5.4.2.7. Módosítás: Írásjelekkel és ékezetek nélkül

Az ékezetek eltávolítását megtartva, úgy gondoltam, hogy az előző adatelemzési adatókból kiindulva (5.2. táblázat) érdemes lehet az írásjeleket is benne hagyni az adatfeldolgozás során. Ezt úgy valósítottam meg, hogy minden mondat közeli- és végi- írásjelet szóközökkel kiegészítettem (például a „?”-ből lett „ ? ”), ha ezt nem végezném el, akkor egy szó ami összekapcsolódik egy írásjellel, például „alma!” az írásjellel együtt egy külön szóként jelenne meg a szókincstáblázatban, ám ebben az esetben az „*alma!*” szóból, „*alma !* „, lesz ami két külön szónak lesz értelmezve. Erről az adatfeldolgozási módszerről készült mérés eredményei az 5.9. táblázatban láthatóak.

5.9. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
Szókincstáblázat mérete (szavak)	7832	-27
Szókincstáblázat mérete (karakterek)	56 837	+578
Szükséges tanító ciklusok száma	24	+10
Pontosság (tanító halmazon)	~99,85%	+0,24%
Pontosság (teszt halmazon)	~83,33%	+3,82%

Az ékezetek továbbra is csökkentik az egyedi szavak számát, így a szókincstáblázat mérete alig változott. A pontosságot tekintve viszont ez a módszer sem tudott pontosabb lenni, mint a sima ékezetek nélküli adatfeldolgozási módszer.

5.4.2.8. Módosítás: Különbségek kiemelése

Az adatelemzésről szóló fejezetben látható volt, hogy az írásjelek használatában, és szóképzésben nagy különbségek vannak a hírek és álhírek között. Ezért az utolsó módosításban arra törekedtem, hogy a szövegfeldolgozási módszer az említett különbségek megtartására koncentráljon. A jelenlegi szövegfeldolgozási módszer minden lépését elemeztem ebből a szempontból.

A kisbetűssé alakítást meg kellett hagynom az 5.4.2.6. módosításban taglaltak miatt. Az írásjeleket meghagyom, de meghatározom, hogy csak bizonyos írásjelek (amiknek a használatában nagy eltérés volt a hírek és álhírek között), maradhatnak egy mondatban, a többi írásjelet pedig eltávolítom. A maradó írásjelek listája (minden írásjel szóközzel elválasztva):

. , - : ! ? %

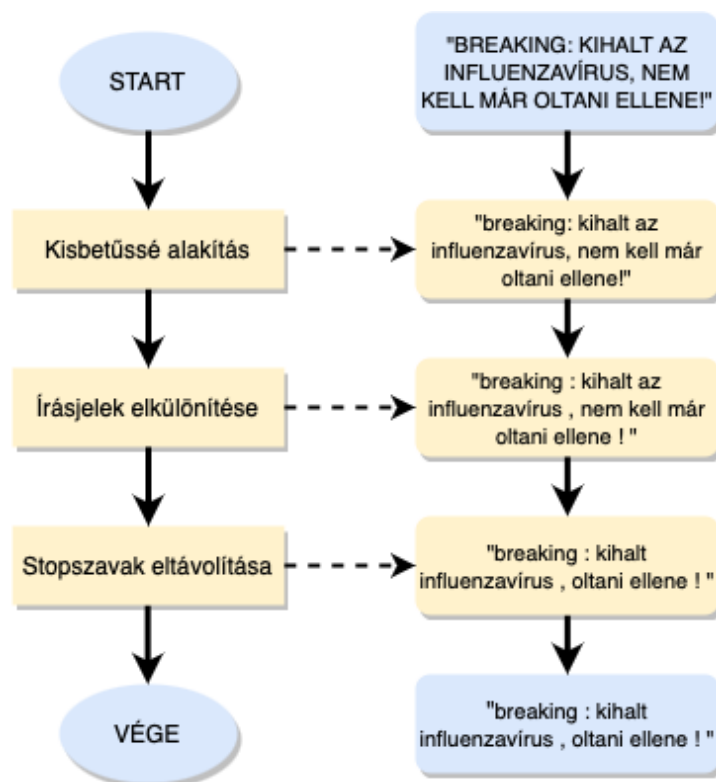
A stopszavakat továbbra is eltávolítom, viszont a szótövezést elhagyom a folyamatból. Az ékezetes karaktereket sem cserélem ki az ékezetmentes karakterpárjukra. Az így kifejlesztett módszerrel mért eredmények az 5.10. táblázatban láthatóak.

5.10. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
Szókincstáblázat mérete (szavak)	10 555	+2696
Szókincstáblázat mérete (karakterek)	88 541	+32 282
Szükséges tanító ciklusok száma	17	+3
Pontosság (tanító halmazon)	~99,95%	+0,34%
Pontosság (teszt halmazon)	~86,24%	+6,73%

Bár a szókincstáblázat mérete nőtt, de a pontosság az eddigi legmagasabb még ezen az egyszerű modellen is, így ez a módszer bizonyul a legsikeresebbnek.

5.4.3. Szövegfeldolgozási módszer megállapítása



5.12. ábra: A szövegfeldolgozó módszer bemutatása, jobb oldalt egy példa hírcímmel

Az előző fejezetben leírt mérések és módosítások után megállapítható, hogy a kutatás szempontjából az eddigi ismeretek alapján legjobb szövegfeldolgozó módszert az utolsó módosítással sikerült elérni, így ezt választottam szövegfeldolgozási módszernek a későbbi fejlesztésekhez. Ennek a módszernek a folyamatábrája látható az 5.12. ábrán és a készülő forráskódról egy képernyőfotó az 5.13. ábrán.

```

def switch_accented(word: str) -> str:
    word = re.sub("[óöő]", "o", word)
    word = re.sub("[úüű]", "u", word)
    word = word.replace("á", "a")
    word = word.replace("é", "e")
    word = word.replace("í", "i")
    return word

def process_sentence(text: str) -> str:
    global stopwords_hu, stemmer
    text = text.lower() # Kisbetűssé alakítás
    regexp = "[^a-zA-Z0-9öüóőúúáúíőüőőúéáúí\ ]"
    text = re.sub(regexp, "", text.strip()) # Írásjelek eltávolítása
    words = [word for word in text.split(" ") if word] # Szavakra darabolás
    words = [word for word in words if word not in stopwords_hu] # Stopszavak eltávolítása
    words = [switch_accented(word) for word in words] # Ékezetes betűk cseréje
    words = [stemmer.stem(word) for word in words] # Szótővezés
    return " ".join(words).strip()
  
```

5.13. ábra: Az adatfeldolgozás metódusai, még a fejlesztési folyamat során, a fent látható metódus kicseréli az ékezetes betűket, az alsó metódus pedig a teljes szövegfeldolgozási folyamatot végrehajtja egy paraméterként kapott mondaton („text” paraméter), mind a két metódus Reguláris Kifejezések használatát tartalmazza

6. Modell fejlesztés és kiértékelés

Ebben a fejezetben bemutatom, hogy milyen mély neurális hálózati modellt fejlesztettem ki a magyar nyelvű álhírek felismerésére, ezen kívül a fejezetben a modellfejlesztés és kiértékelési folyamatot fogom részletesebben ismertetni. Először áttekintem az álhírek felismerésére használható mély tanulási modell típusokat, beleértve a konvolúciós, és visszacsatolt neurális hálózatokat. Ezután részletesen ismertetem, hogy milyen módosításokat hajtottam végre, egy alapnak vett modellen. Végül elemzem az eredményeket, és bemutatom a kutatásom végső modellét.

A modellek tanításához természetesen a felügyelt tanítás [69] módszerét választottam, azaz a modell az adathalmazból vett példákból (álhírek), és ellenpéldákból (hírek) tanulta meg a két hírtípus közötti különbségeket, ami lehetővé teszi, hogy a modell felismerje az álhíreket.

A modelleket az előző fejezetben is használt teszhalmaz módszerrel teszteltem, melynek az a lényege, hogy a teljes adathalmaz 15%-át tesztelésre félrerakom, és a fentmaradó 85%-ot használom fel a modell tanítására. A tesztelésre a későbbi 7. fejezetben részletesebben is kitérek.

6.1. Modelltípusok áttekintése

A modell feladata az, hogy egy mondatnyi szöveget (hírcím) kap bemenetként, amit előzőleg egy adatfeldolgozó módszerrel a modell számára érthetővé tettem, és erről a mondatról meg kell állapítania, hogy mekkora valószínűséggel álhír. Ez egy bináris osztályozási feladat, hiszen két osztály van, ahol az álhírek osztálya 1, a hírek osztálya a 0 számmal lett jelölve. A modell kimenetére $\hat{y}$ -ra igaz, hogy $\hat{y} \in \mathbb{R}$ és $\hat{y} \in [0; 1]$. Ez alapján a feladat alapján kell választani egy olyan modell típust, ami lehetővé teszi, hogy a modell szövegeket osztályozzon, ahol mindössze két osztály van. Ebben a fejezetben ismertetek néhány modell típust, amik erre a feladatra használhatóak.

6.1.1. Teljesen kapcsolt neurális hálózat

Egy olyan neurális hálózati típus, melyben egy réteg összes neuronja, kapcsolódik a következő réteg összes neuronjával. Legtöbbször az ebben található teljesen kapcsolt rétegeket más típusú rétegek mellett használják. Minél több teljesen kapcsolt réteg van egymáshoz kapcsolódva, annál magasabb fokú absztrakciót végez a hálózat, azaz minden következő réteg olyan mintákat

ismer fel, amelyek összetettebbek, mint az előző rétegben [56]. Emellett általában teljesen kapcsolt réteg az utolsó egy neurális hálózatban.

6.1.2. ViSSzacsatolt neurális hálózat

Olyan típusú neurális hálózat/réteg, amelyet a szekvenciális adatokon való tanulásra terveztek, mint a mondatok (ahol szavakból vagy karakterekből állnak össze a szekvenciák), egy videó képkockái, hangfelvétel, vagy az időben változó adatok. A jellegzetessége egy ilyen hálózatnak az, hogy egy belső állapotot tart fenn a bemenetekhez, aminek segítségével az előző bemeneteket is figyelembe veszi a jelenlegin való tanuláskor [60].

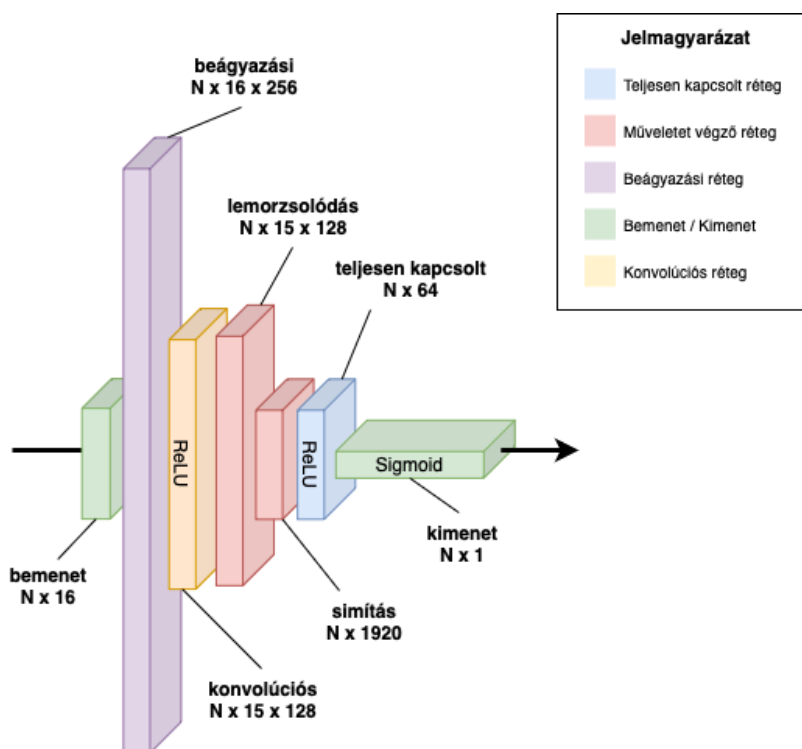
6.1.3. Konvolúciós neurális hálózat

A konvolúciós neurális hálózatokat/rétegeket a képek, és a több mint kétdimenziós adatokon való tanulásra terveztek. A főbb karakterisztikája egy ilyen hálózatnak/rétegnek az, hogy szűrőket használnak a bemeneti adatokon, annak érdekében, hogy különböző mintázattokat fedezzenek fel a bemenetben, aminek segítségével aztán be tudja azonosítani azt. Az ilyen szűrők például a képi adatokon képesek azonosítani a textúrákat vagy egy objektum széleit [55].

6.2. A modell fejlesztése

Ebben a fejezetben bemutatom a kiindulási pontnak használt mély neurális hálózati modellt és a nagyobb módosításokat, amiket eszközöltem rajta. A 6.1. ábrán látható modell volt a kiindulási struktúra [19], a 6.1. táblázatban pedig láthatóak a használt hiperparaméterek. Az alapnak vett modell struktúráját egy, az angol nyelven végrehajtott áhírfelismerésről szóló kutatásból vettem [19]. Mint látható az összes rétegen ReLU [34] aktivációs függvényt használtam, kivéve az utolsón, ami a bináris osztályozási feladat miatt a Sigmoid [35] kell, hogy legyen. A modellt felügyelt tanítással tanítottam az előbbi fejezetekben ismertetett adathalmazon.

Minden módosítás után végzek egy mérést, hogy megbizonyosodjak arról, hogy a módosítás pozitív hatással volt a modell teljesítményére. Fontos megemlíteni, hogy hasonlóan az előző fejezetekben végzett mérésekhez, most is egy statikusan kiválasztott mag számot használtam (ami a negyvenkilenc volt), annak érdekében, hogy a véletlenszerűen generált számok mindig ugyan azok legyenek, így a méréseket a véletlenszerűségtől függetlenül lehet értékelni.



6.1. ábra: Az alapnak vett mély neurális hálózat

Az előző fejezetekben használt modellhez képest, ez a hálózat egy konvolúciós réteget is tartalmaz a teljesen kapcsolt rétegek előtt. Ez a típusú réteg jellemzően a gépi látás (computer vision) [54] területén használt, ám a szövegek bekategorizálásában is pontos eredményeket lehet vele elérni [55].

6.1. táblázat

Hiperparaméter	Érték
Aktivációs függvény	ReLU
Köteg méret	64
Lemorzsolódási ráta	20%
Maximum tanuló ciklusok száma	150
Optimalizációs függvény	Adam (0,001)
Hibafüggvény	Bináris kereszt-entrópia

6.2.1. Előzetes mérések

Még mielőtt változtattam volna az alapnak vett modellen, szükségesnek tartottam, hogy végezzek egy mérést, aminek segítségével meg lehet majd állapítani, hogy a módosítások pozitív vagy negatív hatással voltak a tesztalmazon mért pontosságra. Az alap modelltől készült mérést a 6.2. táblázat ismerteti.

6.2. táblázat

Szempont	Eredmény
RAM igény	47 MB
Pontosság (tanító halmazon)	~68,67%
Pontosság (teszt halmazon)	~57,79%

Az előző fejezetekben ismertetett teszt modell is jobb pontosságot ért el, mint ez a kezdeti modell, ezért tartottam szükségesnek, hogy módosításokat hajtsak végre rajta a pontosság növelésének érdekében.

6.2.2. Lehetőségek a modell javítására

Ahhoz, hogy a kutatáshoz használt alap modell pontosabb legyen, különféle módosítási lehetőségek álltak rendelkezésemre. Ebben az alfejezetben ismertetem a fontosabb lehetőségeket.

6.2.2.1. Adatgyűjtés

A mély tanulási modell felismerési képességére a legnagyobb hatással az adathalmazban található adatok mennyisége és minősége van. Ha a modell gyengén teljesít, akár sok módosítás után is, akkor a nem megfelelő az adathalmaz mérete, vagy a nem megfelelően kategorizált adatok lehetnek a probléma forrásai.

6.2.2.2. Struktúra variálás

A mély neurális hálózatok általában speciálisan megválasztott rétegekkel rendelkeznek, amiknek mind van egy szerepük a hálózatban, ezért egy módosítási lehetőség lehet az, ha vagy a rétegek típusain, vagy a hálózati struktúrán változtatok.

6.2.2.3. Hiperparaméterek finomhangolása

A hiperparaméterek átállítása segíthet a modell teljesítményének javításában. A dolgozatomban használt modellekhez kézzel, és automatizálva is végeztem a finomhangolást. Ehhez felhasználtam az Sklearn szoftverkönyvtárban található rácskeresést (grid search) megvalósító algoritmust.

6.2.3. A modell módosítása

Ebben a fejezetben ismertetem, hogy milyen módosításokat hajtottam végre az alapnak vett modellen, annak érdekében, hogy növeljem a teszhalmazon mérhető pontosságát.

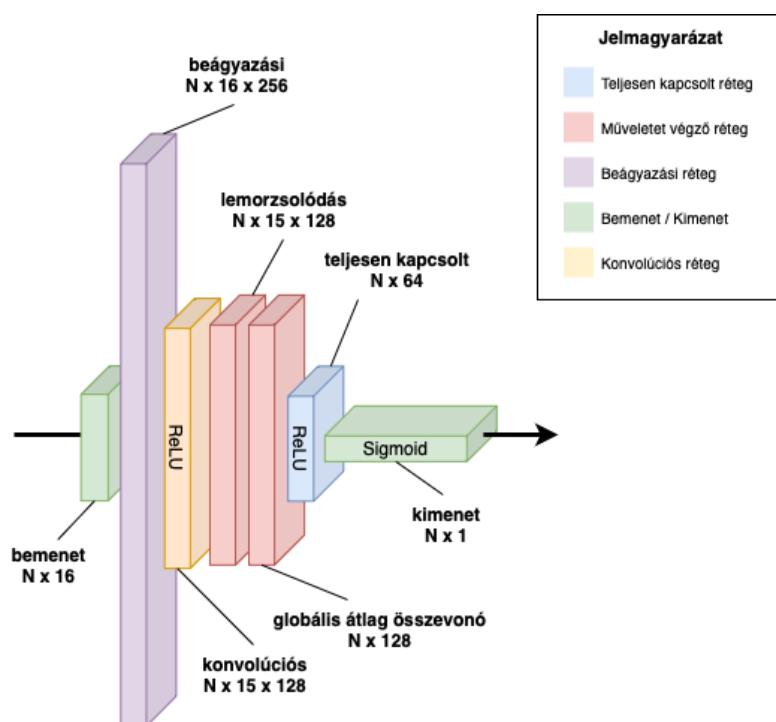
6.2.3.1. Módosítás: Összevonó réteg hozzáadása

Alap konvencionak számít, hogy a konvolúciós réteg után összevonó (pooling) réteg [38] kerül a neurális hálózati struktúrába. Méréseket végeztem annak érdekében, hogy meghatározzam milyen összevonó réteg típust használjak fel a modellben. Az átlag- (average pooling) [39], a maximum- (max pooling) [40] összevonó rétegeket, és ezek globális verzióját is teszteltem. A mérési eredményeket a 6.3. táblázat ismerteti.

6.3. táblázat

Összevonó réteg	RAM igény	Pontosság (tanító halmazon)	Pontosság (teszt halmazon)
Maximum	47 MB	~53,88%	~49,00%
Globális Maximum	47 MB	~71,95%	~58,27%
Átlag	47 MB	~58,36%	~52,78%
Globális Átlag	47 MB	~71,66%	~58,46%

Mint látható az eredményekből, a modell az új rétegek alkalmazása után pontosabb lett, így a struktúrát a 6.2. ábrán láthatóra változtattam, felhasználva egy globális átlag összevonó réteget.



6.2. ábra

6.2.3.2. Módosítás: A modell mérete

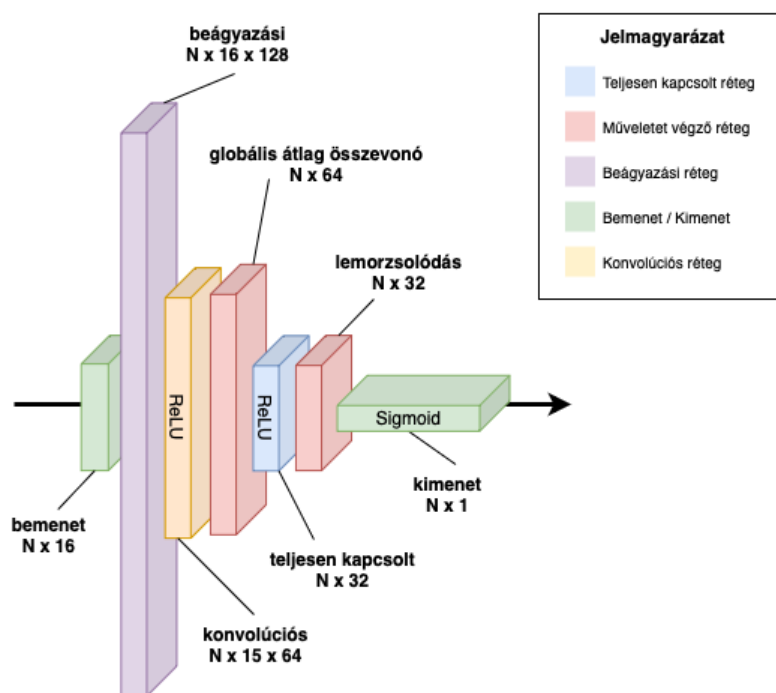
A modell jelenleg 2 079 905 paraméterrel rendelkezik, ami a teljes adathalmaz méretét tekintve nagy szám, így a jobb konvergencia érdekében [46] a rétegek kimeneteit kisebb méretűre állítom (összesen 291 857 paraméterre). Ezen kívül még a lemorzsolódásért felelős réteget

kettővel lejjebb helyeztem a sorban, hogy a teljesen kapcsolt rétegen történjen a lemorzsolódás. A mérés eredménye a 6.4. táblázatban látható.

6.4. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
RAM igény	6 MB	-41 MB
Pontosság (tanító halmazon)	~60,42%	-8,25%
Pontosság (teszt halmazon)	~57,68%	-0,11%

A táblázatból látható, hogy a modell RAM igénye drasztikusan csökkent, ami egy pozitív változás az alaphoz képest, ez annak köszönhető, hogy csökkentettem a rétegek méreteit (paraméterek száma). Pozitív változás az is, hogy a pontosság mind a két esetben csökkent, hiszen a tanító halmazon mért kisebb pontosság azt jelenti, hogy a lemorzsolódásért felelős réteg a helyváltoztatással sokkal jobban kifejti hatását, és ellen tart a túlillesztés problémájának [45]. Ez az állítás különösen megerősödik, ha azt nézzük, hogy a teszhalmazon mért pontosság igen elenyésző mértékben, de csökkent. Az új struktúra szemléltetve van a 6.3. ábrán.



6.3. ábra

Érdekesség, hogy mivel az álhírek jellemzően nagybetűkkel vannak írva (a begyűjtött adatok alapján), ezért, ha az adatfeldolgozás során a kisbetűkre állítás részt kihagyom, akkor a modell nagy mértékben pontosabban azonosítja az álhíreket és híreket. Az ilyen adatfeldolgozással készített méréseket a 6.5. táblázat szemlélteti.

6.5. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
RAM igény	6 MB	-41 MB
Pontosság (tanító halmazon)	~94,15%	+25,48%
Pontosság (teszt halmazon)	~91,56%	+33,77%

6.2.3.3. Módosítás: Szókincstáblázat módosítása

Az adatfeldolgozásról szóló fejezetekben kifejtett és használt számszerűsítési technika egy egyszerű szókincstáblázat építését hajtja végre, ami speciálisan csak az adathalmazban található szókincssel rendelkezik. A szakirodalmi áttekintés után azonban elmondható, hogy a legmodernebb modellek [41] előre elkészített szókincstáblázattal dolgoznak. Ezért kutatásom során felhasználtam a huBERT [4, 29] szókincstáblázatát, ami a Google által készített BERT-hez [42] hasonlóan lett strukturálva. Nagy szókincssel rendelkezik, hiszen részben a magyar Wikipédia [43] tartalmán tanították. Változtatás még az is, hogy az Adam optimalizáló függvény tanuló rátáját 0,001-ről 0,0001-re állítottam, a bemenetek méretét pedig lecsökkentettem 64-re, tehát egy mondat maximum 64 szóból állhat, ha kevesebb szóból áll, akkor maradék szavak helyeit nullásokkal kitöltöm. A mérési eredmények a huBERT használatával a 6.6. táblázatban láthatóak.

6.6. táblázat

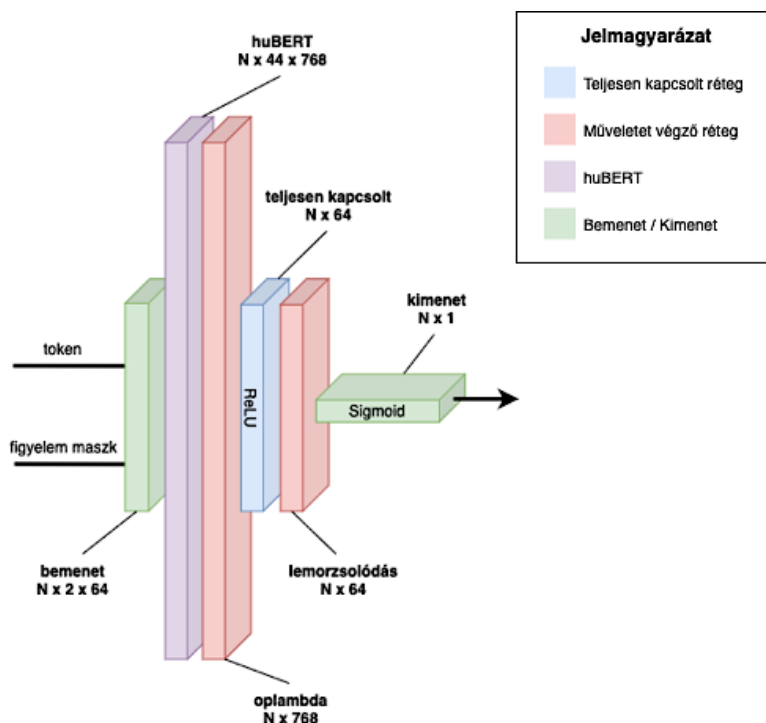
Szempont	Eredmény	Különbség az alaphoz képest
RAM igény	61 MB	+13 MB
Pontosság (tanító halmazon)	~99,90%	+31,23%
Pontosság (teszt halmazon)	~86,00%	+28,21%

A modell RAM igénye megnőtt az előző méréshez képest, mert a maximális szavak számát növeltem, így a hálózat első két rétege több paraméterrel rendelkezik. A pontosság mind a két esetben drasztikusan nőtt, így a továbbiakban is a huBERT szókincstáblázatát fogom használni.

6.2.3.4. Módosítás: Transzfer tanulás

Az előbbieken bemutatott huBERT szókincstáblázathoz tartozik egy előre tanított modell. Amikor egy neurális hálózatba egy másik neurális hálózatot helyeznek, ami már előre volt tanítva általános adatokon, akkor azt transzfer tanulásnak (transfer learning) [47] nevezik. Én egy olyan előre tanított modellt használtam fel, ami a BERT egy magyar nyelvű fejlesztett változata, ami a huBERT [48] névre hallgat. Ebben az alfejezetben bemutatom, hogy ennek az

előre tanított modellnek a segítségével miben változott a kutatás során felhasznált modell. A modell új struktúráját a 6.4. ábra szemlélteti.



6.4. ábra: Látható, hogy a bemeneti réteg most már minden bemenethez kap egy figyelem maszkot is

A magyarázatra szoruló réteg az „oplambda”, ez egy olyan réteg, ami a huBERT-től bemenetként kapott $X \in \mathbb{R}^{N \times 44 \times 768}$ mátrixot $X \in \mathbb{R}^{N \times 768}$ méretűre alakítja egy dimenzió redukció algoritmus [65] segítségével. Ezen kívül az is változás, hogy bemenetként már nemcsak a huBERT szókincstáblázatából származó számokat kapja meg a modell, hanem minden hírcímhez tartozik egy figyelem maszk (attention mask) is, amit a huBERT adat-előfeldolgozója ad kimenetként. A huBERT modellel készült mérések eredményeit a 6.7. táblázat ismerteti.

6.7. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
RAM igény	2533 MB	+2532 MB
Pontosság (tanító halmazon)	~99,90%	+31,23%
Pontosság (teszt halmazon)	~95,33%	+37,54%

Mint látható, ez az eddigi legpontosabb modell, bár a RAM kritériumnak nem felel meg amit a korábbi fejezetekben felállítottam, hiszen több mint két és fél gigabájtnyi memóriát igényel. A transzfer tanulásból készített modellt a pontossága miatt megtartom, és később összevetem egy másik modellel, ami csak a kutatás alatt készített adathalmazon volt tanítva. A továbbiakban a 6.2.3.3. módosítás során elkészített modellel dolgozom tovább.

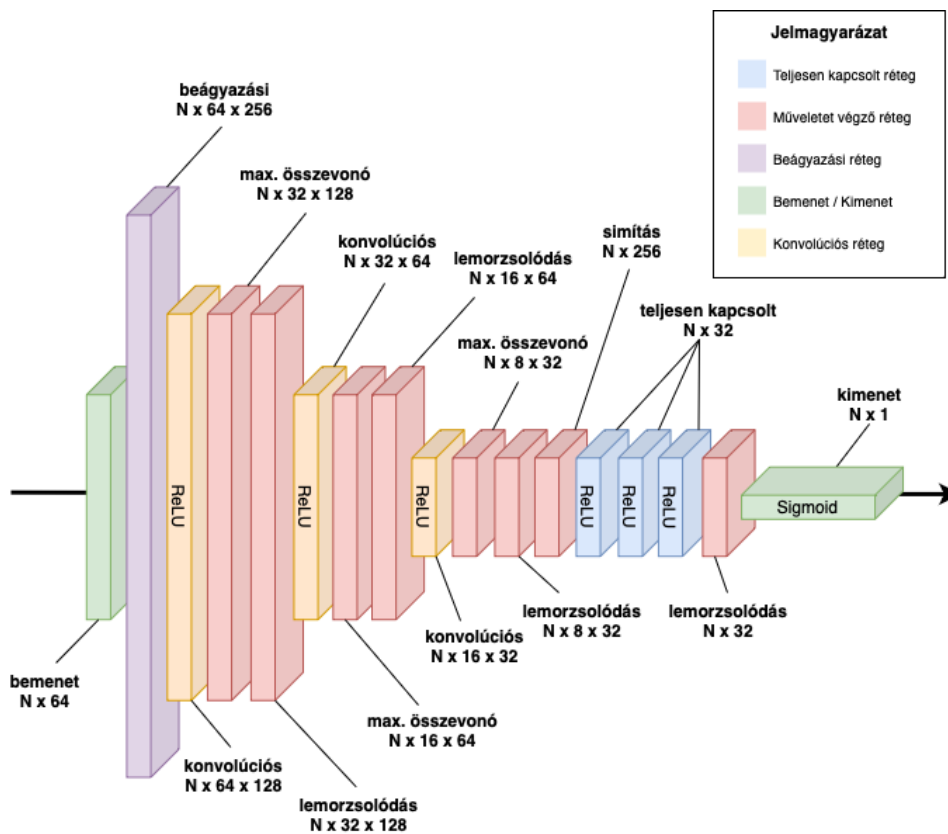
6.2.3.5. Módosítás: Több réteg hozzáadása

Ha több réteg lenne a modellben, akkor bár a paraméterek száma megnőne, ami 6.2.3.2. módosítás ellen dolgozna, de az absztrakció foka nőne [56], ami növelheti a modell felismerési képességét. Három a kimenetek számában fokozatosan csökkenő konvolúciós réteget helyeztem el egymás után. Mindegyik konvolúciós réteg után elhelyeztem egy maximum összevonó- és egy 30%-os lemorzsolódási- réteget. Ezek után a rétegek után egy simító (flatten) [57] réteg helyezkedik el, ami a konvolúciós rétegek háromdimenziós mátrix kimeneteit, „lesimítja” két dimenzióssá. Az egy teljesen kapcsolt réteget a kimeneti réteg előtt, három teljesen kapcsolt rétegre bontottam szét, mindegyik ugyan akkora kimenetet ad. A 6.8. táblázat szemlélteti a módosítás utáni mérés eredményeit.

6.8. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
RAM igény	188 MB	+141 MB
Pontosság (tanító halmazon)	~99,75%	+31,08%
Pontosság (teszt halmazon)	~87,78%	+29,99%

Mint látható, a modell pontossága nőtt, de a túlillesztés mértéke szinte változatlan, így ez a módosítás pozitív hatással volt a hálózatra. Az új hálózati struktúrát a 6.5. ábra szemlélteti.



6.5. ábra

6.2.3.6. Módosítás: Aktivációs függvények módosítása

Az haldokló ReLU (dying ReLU) [58] problémájának elkerülése érdekében a jelenleg ReLU aktivációs függvényeket használó rétegeknél, átállítottam az aktivációs függvényt LeakyReLU-ra [59], $\alpha = 0,3$ értékkel. A 6.9. táblázat szemlélteti a mérések eredményét.

6.9. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
RAM igény	189 MB	+142 MB
Pontosság (tanító halmazon)	~100,00%	+31,33%
Pontosság (teszt halmazon)	~83,11%	+25,32%

A modell pontatlanabb ezzel az aktivációs függvénnyel, a túlillesztés pedig továbbra is magasfokú a kezdeti állapothoz képest. Ezért az aktiváló függvényeket visszaállítom ReLU-ra.

6.2.3.7. Módosítás: Visszacsatolt réteg hozzáadása

A természetes szövegekkel dolgozó neurális hálózatok nagy része visszacsatolt rétegeket (recurrent layer) [60] használ. Ezekből is az LSTM (long-short term memory) [61] rétegek azok, amik hosszú- és rövid- távú memóriával rendelkeznek, ami egy neurális hálózat esetén hasznos lehet, ha mondatonként dolgozik (például: versnek szánt szöveg generálásnál emlékeznie kell a hálózatnak az előző sorra, hogy attól eltérő legyen a következő generált sor [62]). A kutatásomhoz leginkább használható visszacsatolt réteg a GRU [63], ami az LSTM-hez hasonlóan működik, azonban kevesebb memóriát használ, és gyorsabb az olyan rövidebb szövegeken, mint a hírcímek [64]. Tehát a simító réteg elé helyeztem egy GRU réteget, hogy így is letegyem a modell pontosságát. Ezen kívül ezzel a módosítással csökkentettem a rétegek méreteit, az előző módosítás során felmerülő túlillesztés csökkentése érdekében. A módosított modellel készített mérések eredményeit a 6.10. táblázat szemlélteti.

6.10. táblázat

Szempont	Eredmény	Különbség az alaphoz képest
RAM igény	188 MB	+141 MB
Pontosság (tanító halmazon)	~99,8%	+31,13%
Pontosság (teszt halmazon)	~86,89%	+29,10%

Mivel a mérések azt mutatják, hogy a modell pontatlanabb a 6.2.3.5. módosítás során megállapított modellnél. Így ezt a módosítást is visszavonom, és tovább folytatom a finomításokat a 6.5. ábrán látható modellel.

6.3. A modell finomítása

Ahogy a 6.2.2. fejezetben is említettem, a pontosság javítására megoldás lehet egy nagyobb adathalmaz. Mivel a jelenlegi adathalmaz kb. 3000 hírcímmel rendelkezik, így célul tűztem ki, hogy gyűjtssek még 5000 hírt és 5000 álhírt, nem beleszámítva az előző adathalmaz méretét. Ebben a fejezetben bemutatom, hogy hogyan gyűjtöttem, szűrtem, majd tanítottam a modellt ezekkel az új adatokkal, hogy egy pontosabb modellt kaphassak eredményül.

Fontos megjegyezni, hogy a modellt már betanítottam és lementettem az ebben a fejezetben bemutatott folyamatok előtt. Így a finomításhoz való adatgyűjtés után, csak az új adatokon tanítottam tovább a modellt.

6.3.1. Adatgyűjtés

A kitűzött 10 000 begyűjtendő hírcím többszöröse a jelenlegi adathalmaz méretének, ezért a gyűjtéshez kizárólag az 5.1.2. fejezetben bemutatott automatizált gyűjtési módszert alkalmaztam. Az alap szűrési algoritmust itt is használtam, és a hírcímek kategorizálása ugyan úgy a listányi webcímek alapján történt. Természetesen arról már a gyűjtési folyamat során megbizonyosodtam, hogy az új adathalmaz ne tartalmazzon olyan hírcímeket, amik szerepelnek a régi adathalmazban. Erre a halmazra a továbbiakban, mint „az új adathalmaz”, az előzőekben használt adathalmazra pedig mint „a régi adathalmaz” fogok hivatkozni.

6.3.2. Adatszűrés

Mivel az adatokat automatizálva gyűjtöttem be, így szükségesnek tartottam, hogy megsűrjem az új adathalmazt, arra figyelve, hogy a hírcímek a megfelelő kategóriákban vannak-e. Erre a feladatra viszont alkalmatlan a kézi szűrés, hiszen túl sok időt venne el 10 000 hírcím végig olvasása és értelmezése, ezért egy új szűrési módszert fejlesztettem ki erre a feladatra.

6.3.2.1. Modellel szűrés

Az adatok szűréséhez felhasználtam az előzőleg tanított modellt (6.5. ábra), és az adatokon végrehajtottam a következő algoritmust:

6.1. algoritmus

```

1  FÜGGVÉNY MODELLEL_SZŰRÉS(HÍRCÍM, KATEGÓRIA)
2      ÁLHÍR_VALÓSZÍNŰSÉGE := MODELL_SZERINT(HÍRCÍM)
3      HA ÁLHÍR_VALÓSZÍNŰSÉGE >= 0,6 ÉS KATEGÓRIA = 1 AKKOR
4          VISSZATÉRÉS MENTÉS_ÁLHÍRKÉNT(HÍRCÍM)
5      ELÁGAZÁS VÉGE
6      HA ÁLHÍR_VALÓSZÍNŰSÉGE <= 0,4 ÉS KATEGÓRIA = 0 AKKOR
7          VISSZATÉRÉS MENTÉS_HÍRKÉNT(HÍRCÍM)
8      ELÁGAZÁS VÉGE
9      VISSZATÉRÉS MONDAT_TÖRLÉSE(HÍRCÍM)
10  ELJÁRÁS VÉGE

```

A begyűjtés során már megtörtént a bekategorizálása a hírcímeknek, ezért a függvény második paraméterként kapja a **KATEGÓRIA** paramétert, ami tartalmazza a hírcím eredeti kategóriáját (ha **KATEGÓRIA = 1** akkor álhír, de ha **KATEGÓRIA = 0** akkor hír a **HÍRCÍM** alap kategóriája). Az algoritmus először azt csinálja, hogy megállapítja a modellel az adott hírcímhez köthető álhír valószínűségét. Tehát a modell bemenetként kapja a hírcímet (**MODELL_SZERINT** függvény), és a kimenetet lementem az **ÁLHÍR_VALÓSZÍNŰSÉGE** nevű változóba. Ezután szortíroz a következőképpen:

- Ha az adott hírcím a modell szerint **60% vagy a feletti** bizonyossággal álhír és a kategóriája **álhír**, akkor **álhírként** mentem a hírcímet (**MENTÉS_ÁLHÍRKÉNT**).
- Ha az adott hírcím a modell szerint **40% vagy az alatti** bizonyossággal álhír és a kategóriája **hír**, akkor **hírként** mentem a hírcímet (**MENTÉS_HÍRKÉNT**).
- Ha egyik kategóriába sem szortíroztam eddig az adott hírcímet (azaz, ha $0,4 < \hat{y} < 0,6$ ahol $\hat{y}$ az álhír valószínűsége a modell szerint), akkor törlöm azt az adatbázisból (**MONDAT_TÖRLÉSE**).

Az ezzel a módszerrel megszürt új adathalmaz kevesebb mint 10 000 hírcímből állt, így további automatizált gyűjtés és modellel szűrés volt szükséges ahhoz, hogy az újonnan beszerzett adathalmaz 5000 hírből és 5000 álhírből álljon.

Annak érdekében, hogy megbizonyosodhassak arról, hogy a modell ténylegesen csak a szükségtelen hírcímeket szűrte ki, a kiszűrt híreket és álhíreket egyaránt ellenőriztem a 5.2.2. fejezetben bemutatott kézi szűrés segítségével. A 0,4 és 0,6 határvonalakat úgy állapítottam meg, hogy többféle határvonallal is kísérleteztem (0,05 mértékkel), és ezeket találtam a modell pontosságát és megbízhatóságát (varianciateszt alapján, 7.3.2. fejezet) figyelembe véve az optimálisnak.

6.3.2.2. Szűrés kiértékelése

A modellel való szűrés lehetővé tette, hogy az új adathalmazból csak az olyan hírcímeket tartsam meg, amelyekben a jelenlegi betanított modell biztos a kategorizálás terén (ha $0,4 < \hat{y} < 0,6$ esetet bizonytalanak veszem), így ezeket a hírcímeket nem szűrtem tovább más módszerrel. A szűrés utáni adathalmazról néhány alap adatot ismertet a 6.11. táblázat.

6.11. táblázat

\	Adathalmaz mérete	Leghosszabb mondat	Mondatok átlag hossza	Leggyakoribb mondat végi írásjel
Hírek	4979 hírcím	155 betű / 23 szó	56 betű / 8 szó	?
Álhírek	4997 hírcím	281 betű / 41 szó	81 betű / 12 szó	!

6.3.3. A modell tovább tanítása

Miután felállítottam az új adathalmazt, a meglévő és már a régi adathalmazon tanított modellt tovább tanítottam ezen az új halmazon. Ugyan úgy ment a tanítás, mint az előző fejezetekben is. Az egyetlen különbség az, hogy az új adathalmazt, ami 9 976 hírcímből áll, nem 85-15 arányban, hanem 80-20 arányban osztottam fel tanító- és tesztelő- halmazokra. Tehát a tanulóhalmaz 7 979, a tesztelőhalmaz pedig 1 997 hírcímből állt. A tovább tanítás során nem használtam fel a régebbi adathalmazt, amin a modell előzőleg tanult, hiszen akkor az eredmények csupán a túlillesztés miatt lennének pontosabbak, mint eddig. A tanulási folyamat utáni mérések eredményei a 6.12. táblázatban vannak ismertetve.

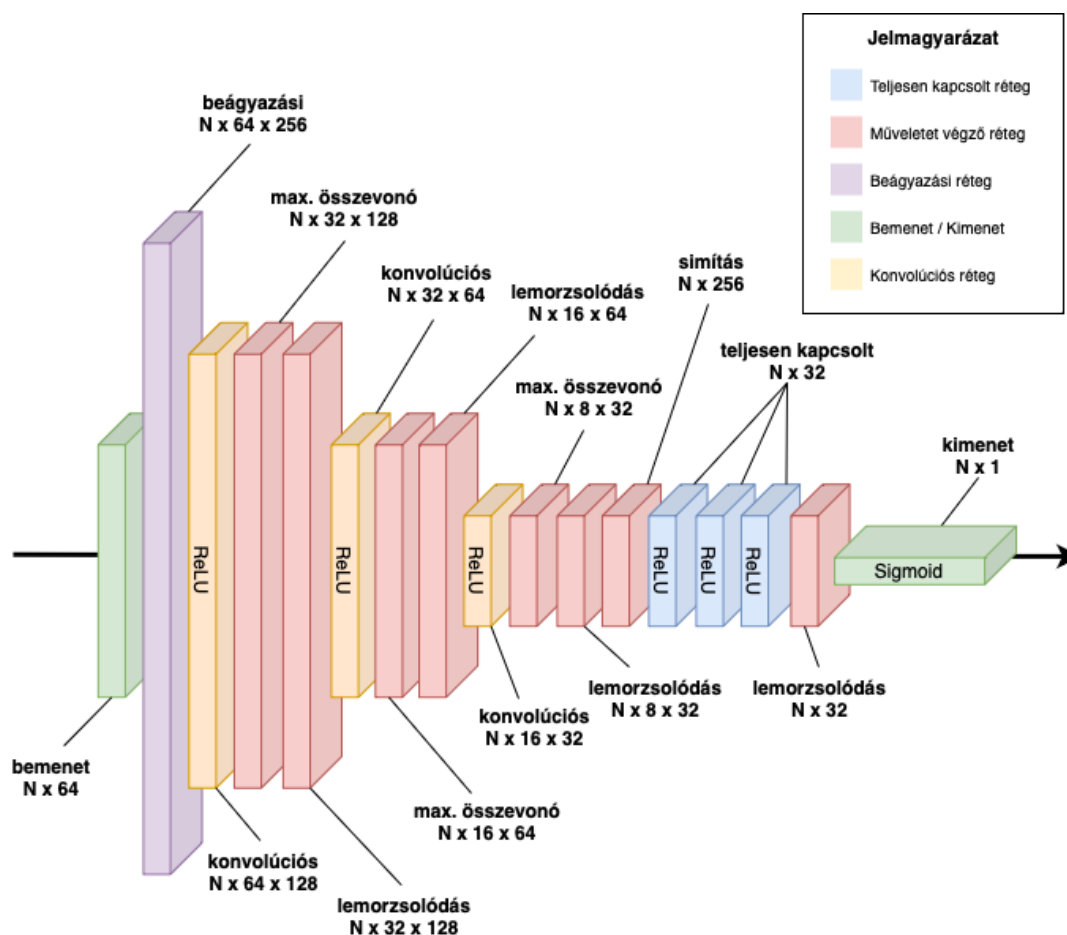
6.12. táblázat

Szempont	Eredmény
Pontosság a tanító halmazon	~99,77%
Pontosság a teszt halmazon	~97,23%

Mint látható a modell pontossága drasztikusan nőtt, a túlillesztés mértéke pedig az előző módosítások során mértekhez képest nem túl magas.

6.4. A modell megállapítása

Ebben a fejezetben minden fontosabb paraméterét ismertetem a kutatás során kifejlesztett legjobb modellnek. Ezzel lehetővé teszem, hogy az eredményeket megismételhesse bárki, aki hozzáfér a dolgozatomhoz. A modell végleges struktúráját a 6.6. ábra szemlélteti.



6.6. ábra

Az egész programra az 88-as szám volt magnak beállítva. Az optimalizációs függvény maradt a már említett Adam algoritmus 0,0001-es tanítási rátával, ehhez hibafüggvénynek bináris kereszt entrópia volt használva. A tanítási ciklusok száma maximalizálva volt 150-re, a köteg méret 64-re, és validációs adathalmaz mérete 20%-ra és véletlenszerű kiválasztásra volt beállítva (az előbb is említett mag számmal).

```
init_model = lambda: tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=total_words+1, output_dim=256, input_length=maxlen),

    tf.keras.layers.Conv1D(128, kernel_size=2, padding="same", activation="relu"),
    tf.keras.layers.MaxPooling1D(),
    tf.keras.layers.Dropout(0.3, seed=SEED),

    tf.keras.layers.Conv1D(64, kernel_size=2, padding="same", activation="relu"),
    tf.keras.layers.MaxPooling1D(),
    tf.keras.layers.Dropout(0.3, seed=SEED),

    tf.keras.layers.Conv1D(32, kernel_size=2, padding="same", activation="relu"),
    tf.keras.layers.MaxPooling1D(),
    tf.keras.layers.Dropout(0.3, seed=SEED),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(32, activation="relu"),
    tf.keras.layers.Dense(32, activation="relu"),
    tf.keras.layers.Dense(32, activation="relu"),
    tf.keras.layers.Dropout(0.5, seed=SEED),

    tf.keras.layers.Dense(1, activation="sigmoid")
])
```

6.7. ábra: A modell inicializálásához írt forráskód, a kép még a fejlesztési folyamat során készült

A teszhalmaz 15%-át, a tanító halmaz pedig 85%-át vette el a teljes halmaznak, majd pedig a finomítási folyamat során 80-20 arányban voltak felosztva az újonnan beszerzett adatok. Az adatfeldolgozási technika az 5.4. fejezetben volt ismertetve. A tanítás során a korai leállás módszerét alkalmaztam, tíz tanító ciklusnyi türelmi idővel.

6.13. táblázat

Kategória	Pontosság
Hír	96,92%
Álhír	97,54%

Hasonlóan a 3.1. táblázathoz, a 6.13. táblázat is ismerteti a pontosságot a teszhalmazon, kategóriánként.

```
model.compile(
    metrics=["accuracy"],
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-04),
    loss=tf.keras.losses.BinaryCrossentropy()
)

early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_accuracy',
    mode='max',
    patience=10,
    verbose=False,
    restore_best_weights=True
)

history = model.fit(
    X_train, y_train,
    epochs=150,
    validation_split=0.2,
    batch_size=64,
    callbacks=[early_stop],
    verbose=True
)
```

6.8. ábra: A modell tanításához írt forráskód, ezen látható a hibafüggvény, az optimalizáló és a korai leállás is

A modellel készített mérések eredménye megtekinthető a 6.14. táblázatban, a dolgozathoz véglegesnek vett pontossági mérések pirossal vastagon kiemelve láthatóak rajta.

6.14. táblázat

Szempont	Eredmény
RAM igény	188 MB
Pontosság a régi tanítóhalmazon, finomítás előtt	99,75%
Pontosság a régi teszhalmazon, finomítás előtt	87,78%
Pontosság az új tanítóhalmazon, finomítás után	99,77%
Pontosság az új teszhalmazon, finomítás után	97,23%
Szükséges tanító ciklusok száma	13
Szükséges tanító ciklusok száma a finomításhoz	13

A 6.15. táblázat pedig szemlélteti a transzfer tanulással készített modell mérési eredményeit.

6.15. táblázat

Szempont	Eredmény
RAM igény	2533 MB
Pontosság (a régi tanítóhalmazon)	99,90%
Pontosság (a régi teszhalmazon)	95,33%
Szükséges tanító ciklusok száma	12

A tanítási folyamat szemléltetése érdekében, készítettem és megtekinthetővé tettem egy jegyzetfüzetet [83]³.

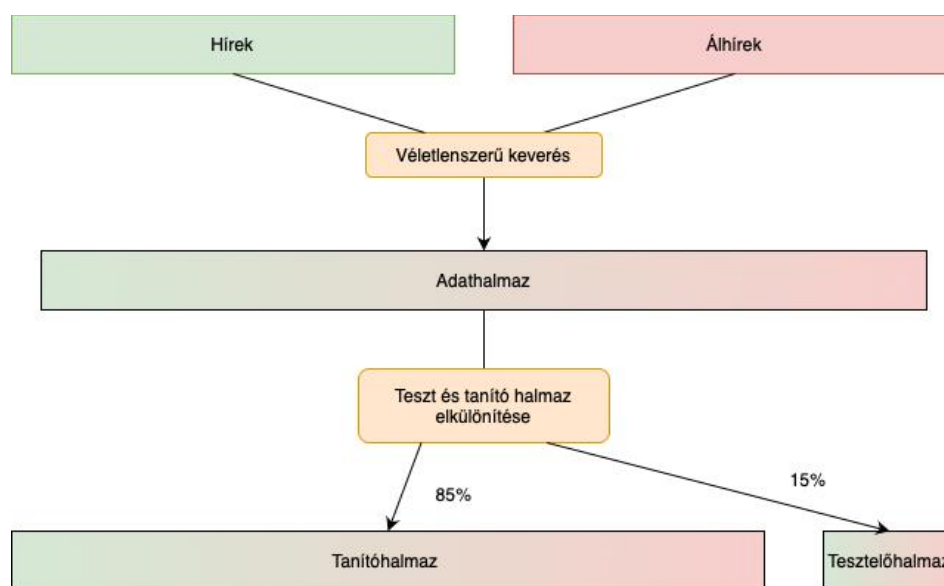
³ Eltéréseket tartalmazhat a dolgozatban szereplő adatoktól (mint az adathalmaz mérete, tanítóciklusok száma stb.), ennek oka az, hogy a jegyzetfüzet a legfrissebb eredményeket tartalmazza, így ezt a dolgozat leadása után is folyamatosan frissítem.

7. Modellek tesztelése

Ebben a fejezetben bemutatom, hogy milyen módon teszteltem a kutatás során a modelleket, illetve, hogy a legjobb modell milyen eredményeket ért el ezeken a teszteken. Ahogy a dolgozat adatelemzés részénél is, így ebben a fejezetben is a finomítási folyamat előtt gyűjtött adatbázison végeztem a tesztek.

7.1. Teszthalmaz módszer

A kutatásom során legtöbbször ezt a módszert használtam a modellek pontosságának mérésére. Első dolgom volt az adathalmazt soronként véletlenszerűen megkeverni, mert a soronkénti megkeverés nem állítja át az adott hírcímhez tartozó osztályt (álhír vagy hír). Ezek után még a feldolgozás előtt két halmazra szedtem szét az egész adathalmazt. Az egyik halmaz volt a tanító-, a másik pedig a tesztelő- halmaz. A teljes adathalmazt a konvenció szerint 80-20 arányban osztják el, ahol 80% a tanító- és 20% a tesztelő- halmaz. A kutatás során a teszthalmaz a teljes adathalmaz 15%-át tette ki, így a tanításra megmaradt 85%. Azért választottam a 15-85 számokat és nem a 20-80-at, mert a tanító halmaz kicsinek számít, a népszerűbb szövegeket tartalmazó adathalmazok között [66] (kb. 2500 hírcím), így a tanításra több elemet hagytam meg mint 80%, ami kb. 2350 hírcím lenne. Az adathalmazokat ezután külön kell kezelni, és a teszt halmazon a modell nem végezhet semmilyen műveletet amíg a tanítási fázis aktív [28]. Ennek a folyamatnak a szemléltetésére látható a 7.1. ábrán.



7.1. ábra

Miután a tanítási folyamat lefutott, a teszhalmazban található hírcímeken lefuttatom az adatfeldolgozást, és a modellel megállapítatom minden hírcímre az álhír valószínűségét. Azon hírcímek esetében, ahol a valószínűség 50%, vagy a felett van, álhírnek tekintem.

Ezek után veszem a hírcímek modell által feltételezett kategóriáját $\hat{c}_i \in \{0; 1\}$ (hír = 0, álhír = 1), és a valódi kategóriájukat $c_i \in \{0; 1\}$ ahol $i = \{1, 2, 3, \dots, N - 1, N\}$ (ahol N a teszhalmaz hírcímeinek száma) és elvégzem a következő algoritmust:

7.1. algoritmus

```

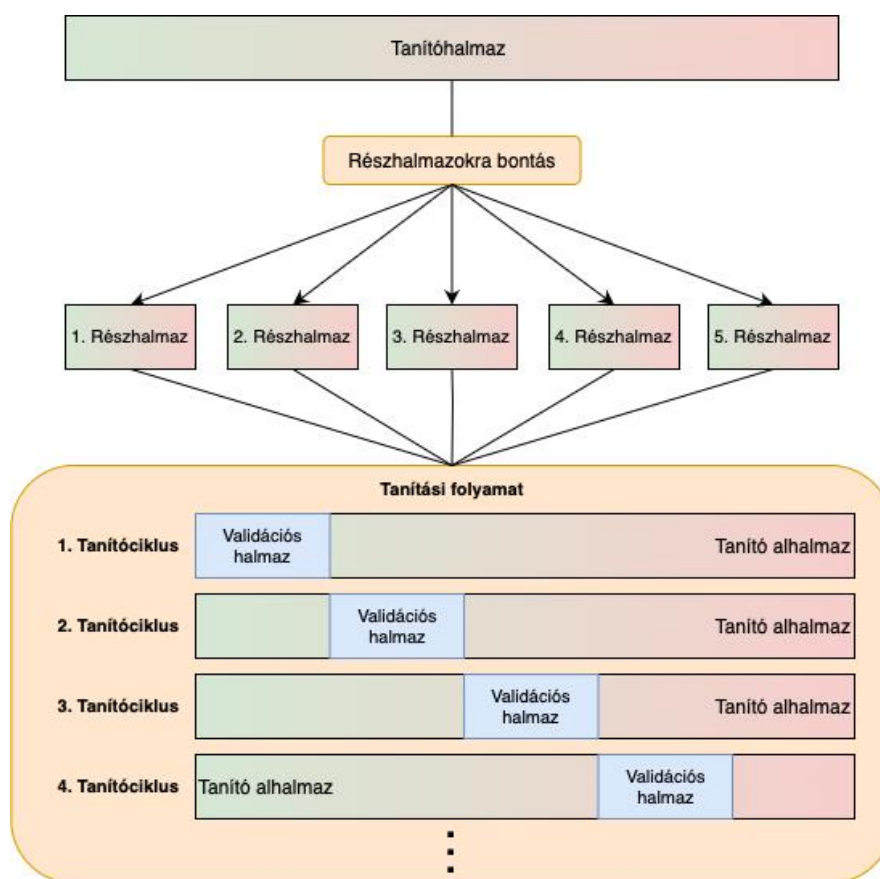
1  FÜGGVÉNY PONTOSSÁG(C, C_KALAP, N)
2      P := 0
3      CIKLUS I = 0-TÓL (N-1)-IG 1-EL
4          HA C[I] = C_KALAP[I] AKKOR
5              P := P + 1
6          ELÁGAZÁS VÉGE
7      CIKLUS VÉGE
8      VISSZATÉRÉS (100 * P / N)
9  FÜGGVÉNY VÉGE

```

Ami kiszámolja, és visszatérési értéként adja a modell százalékos pontosságát a teszhalmazon. A kutatás során minden pontossági érték kiszámításához ezt az algoritmust alkalmaztam.

7.2. K-szoros keresztvalidációs módszer

A tanítási folyamat során korai leállást alkalmaztam, ezért szükség volt egy mértékre, amivel követhetem a modell pontosságát a tanítóciklusok futása közben. Erre a K-szoros keresztvalidációs- (K-fold cross-validation) [67] vagy egyszerűen a validációs halmaztechnikát választottam. A validációs halmazt a tanítóhalmazból kell származtatni úgy, hogy egy előre meghatározott méretű halmazokra felbontja azt (a kutatás esetében $K=5$), és minden tanítóciklus kezdetekor kiválaszt egy részhalmazt és kinevezi azt validációs halmaznak, a fennmaradó részhalmazokat pedig tanító alhalmaznak. Ezek után a modell a tanító alhalmazon elvégzi a tanítási folyamatot, és a tanító ciklus végén a validációs halmazon tesztet futtat le. A folyamat vizualizálva látható a 7.2. ábrán.



7.2. ábra

Mint ahogy az ábrán is látható, a validációs halmaz minden tanító ciklus során változik, és egy másik részhalmaz lesz kiválasztva erre a feladatra. Ha már az összes részhalmaz volt validációs halmaz a tanítás során, akkor a sorrend előlről kezdődik. A validációs halmazon végzett teszt pontossága volt az a mérték, amely alapján a korai leállás el tudta dönteni, hogy mikor kell félbeszakítani a tanítási folyamatot.

7.3. Egyéb tesztek a tanítás után

Ahhoz, hogy megbizonyosodhassak róla, hogy a modell valóban megfelelő pontossággal működik, további tesztek futtatam le vele. A tesztek bemutatam, és a legjobb modellen végzett eredményeket ismertetem ebben a fejezetben.

7.3.1. Pontosság táblázatba foglalása

Írtam egy programot, ami készít egy Excel [68] programmal megnyitható táblázatot. A táblázat a következő oszlopokat tartalmazza: hírcím, kategória, álhír valószínűsége, kimenet. A teljes adathalmazon futtatam a modell felismerésért felelős kódot és az eredményeket ebbe a

táblázatba mentettem el. A kimenet oszlopban valós számok vannak, amiket a modell az adott sorban található hírcímhez állapított meg. A táblázatról látható egy képernyőfotó a 7.3. ábrán.

HÍRCÍM	KATEGÓRIA	ÁLHÍR VALÓSZÍNŰSÉGE (%)	KIMENET
Egy feledékeny effektus buktatta le a sárkányos királyt	0	0,89%	0.008866140618920326
Drei Zinnen: ahol az égbe meredő sziklakarmok a háborút idézik	0	41,58%	0.4158378541469574
Miért játszik spanyol focibajnokságokban két afrikai csapat is?	0	0,60%	0.005957321729511023
Mi köze Sir David Attenborough-nak egy viktoriánus kori gyilkossági rejtélyhez?	0	12,79%	0.12787578999996185
Az ír szent, aki sörré változtatta a fürdővizet	0	0,22%	0.0022077676840126514
A vonat, ami egy 19 emeletes lakóépületen zakatol át Kínában	0	0,52%	0.005247135646641254
Mégis hogy működik a szappan?	0	1,03%	0.010290371254086494
Ha baj van,ők kétszázzal szállítják a vért	0	4,33%	0.043311707675457
Le lehet lőni egymás után egy csomó medvét, de ez nem hoz megoldást	0	0,67%	0.0066629196517169476
Gránát végzett a jobb kezével, ezért ballal nyerte az olimpiákat	0	0,33%	0.0032690612133592367
„Ha van bárkinek más ötlete az elégedetlenség kifejezésére, szívesen meghallgatjuk”	0	1,22%	0.012173067778348923
Bayer Zsolt: Egy lator államért kellene megfagynunk?	0	2,32%	0.023152688518166542
Erzsébetvárosi előterjesztés: októbertől a hét öt napján éjjelkor bezárhat minden vendéglátóhely	0	6,53%	0.06528597325086594
Szállodák zárhatnak be, elbocsátások jöhetnek	0	0,78%	0.007827027700841427
Holtan találtak egy nőt Nógrádban, megölhették	0	0,33%	0.0032782736234366894
„A Vodafone fontosabb, mint a pedagógusbér-emelés”	0	0,35%	0.0035448691342025995

7.3. ábra

7.3.2. Variancia teszt

Ezen kívül variancia tesztet végeztem, hogy megállapítsam a modell eredményeit mennyiben befolyásolja a változás a bemenetekben. Kezdetnek vettem egy kitalált hírcímet „Megtalálták a kincset, amit kerestek”. Ezt alap esetben egyszerű hírként kéne, hogy vegye modell. Az eredményt a 7.1. táblázat ismerteti.

7.1. táblázat

Hírcím	Álhír valószínűsége	Feltételezett kategória
Megtalálták a kincset, amit kerestek	10,61%	Hír

A modell helyesen osztályozta a hírcímet hírnek. A továbbiakban a fenti táblázatban látható szempontok alapján fogom vizsgálni a varianciát, változó példa hírcímekkel.

7.3.2.1. Mondat végi írásjel variálása

Ha a bemenetet mondat végi írásjeleit variálom, akkor változást tapasztalok a kimenetben is. A mondat végi írásjelek variálásával végzett tesztek eredményei a 7.2. táblázatban láthatóak.

7.2. táblázat

Hírcím	Álhír valószínűsége	Feltételezett kategória
Megtalálták a kincset, amit kerestek!	58,80%	Álhír
Megtalálták a kincset, amit kerestek!!	90,96%	Álhír
Megtalálták a kincset, amit kerestek!!!	98,68%	Álhír

Az eredmények alapján a felkiáltójel növelte a modell számára az álhír valószínűségét. A következő, 7.3. táblázat ismerteti a kérdőjellel végzett hasonló tesztek eredményeit.

7.3. táblázat

Hírcím	Álhír valószínűsége	Feltételezett kategória
Megtalálták a kincset, amit kerestek?	20,13%	Hír
Megtalálták a kincset, amit kerestek??	63,92%	Álhír
Megtalálták a kincset, amit kerestek???	93,66%	Álhír

Ahogy az előző két táblázat is mutatja, minél több mondat végi írásjel van egy hírcím végén, a modell annál biztosabb benne, hogy egy álhírt kapott bemenetként. A felkiáltójel esetén, meredek emelkedés látható ahogy a felkiáltójelek száma nő (59%, 91%, 99%), ellenben azzal az esettel amikor a mondat végi írásjel kérdőjel, és a valószínűség lassabban nő (20%, 64%, 94%).

7.3.2.2. Szókészlet variálása

Mivel a modell a szavak használata alapján dönti el, hogy a hírcím, amit kapott álhír-e. Így érdemesnek tartottam ellenőrizni, hogy vajon milyen szavakra, hogyan változik a valószínűség. Ennek a tesztnek az eredményeit szemlélteti a 7.4. táblázat.

7.4. táblázat

Hírcím	Álhír valószínűsége	Feltételezett kategória
Vihar közeleg Magyarországra	36,13%	Hír
Nagy vihar közeleg Magyarországra	36,13%	Hír
Hatalmas vihar közeleg Magyarországra	63,20%	Álhír

Mint látható, egy hírből „Vihar közeleg Magyarországra”, álhír lett a modell szerint, a „hatalmas” szó használata után. Ennek köze lehet ahhoz a tényhez, hogy az adatbázisban található álhírek gyakrabban használnak túlzó szavakat, mint a hírek.

7.3.2.3. Fogalmazás variálása

Kipróbáltam, hogy a modell hogyan reagál egy hírre akkor, ha ugyan azt a hírt átfogalmazom háromféleképpen, a hírcím jelentése egyik esetben sem változik. Az eredményeket a 7.5. táblázat ismerteti.

7.5. táblázat

Hírcím	Álhír valószínűsége	Feltételezett kategória
Csökkentek a nagyvárosi lakásárak országszerte	10,79%	Hír
A nagyvárosokban csökkentek a lakásárak	10,61%	Hír
A magyar nagyvárosokban kicsit már csökkentek is a lakásárak	10,33%	Hír

Látható, hogy a modellt nem tévesztette meg a fogalmazás variálása, mert közel azonos valószínűséget jósolt minden hírcímre a táblázatban.

7.3.2.4. Szórakoztatási célzattal írt hírek

Vincze Veronika a *Kacsa vagy nem kacsa? Magyar nyelvű álhírek automatikus azonosítása* [2] című kutatásban, hasonlóan a dolgozatomhoz az álhírek azonosításával foglalkozik. Az említett kutatásban az álhírhalmazt szórakoztatási célból megírt hírek tették ki, így indokoltnak láttam tesztelni, hogy a dolgozatomban fejlesztett modell, a szórakoztatási célzattal írt híreket milyen kategóriába sorolja. A teszt eredményei ismertetve vannak a 7.6. táblázatban.

7.6. táblázat

Hírcím	Álhír valószínűsége	Feltételezett kategória
Zuckerberg megvette Magyarországot	35,14%	Hír
Hamis rokkant kártyával csalt el fiatal nőket	11,43%	Hír
Börtönbüntetés fenyegeti a hatoslottó nyertesét	11,35%	Hír

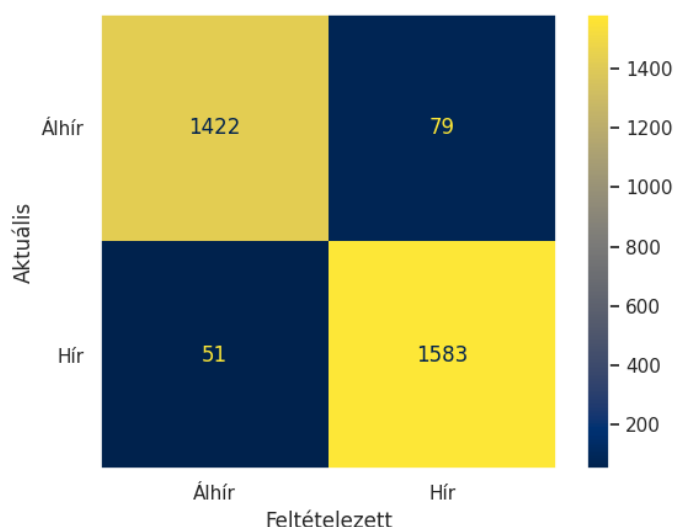
Mint látható a táblázatból, a modell a szórakoztatásból megírt híreket legtöbbször egyértelműen a hírek kategóriájába sorolja.

7.3.3. Zavarmátrix

A zavarmátrix (confusion matrix) [70] egy olyan táblázat, amit gyakran használnak a gépi tanulási modellek osztályozási teljesítményének ellenőrzésére. Két ismétlődő kifejezést láthatunk a 7.4. ábrán bemutatott zavarmátrixon, az egyik a „Hír”, a másik az „Álhír”. A zavarmátrix egy négy részre osztott ábra, aminek részeit a következőkben ismertetem.

Az „Álhír-Álhír”, más néven, igaz-pozitív (bal felső sarok) azoknak az elemeknek a számát tartalmazza, amik álhírek, és a modell is álhíreknek állapította meg őket. A jobb alsó

sarokban található számhoz tartozó „Hír-Hír” (igaz-negatív) azt jelenti, hogy a modell ennyi hírt állapított meg hírnek, amik valóban azok voltak. A jobb felső sarokban, az „Álhír-Hír” (hamis-negatív) azoknak a hírcímeknek a számát tartalmazza, amik valójában álhírek, de a modell szerint hírek. Végül a bal alsó sarokban látható „Hír-Álhír” (hamis-pozitív) azoknak a hírcímeknek a száma, amik valójában hírek, de a modell szerint álhírek.



7.4. ábra

Ahogy láthatjuk a táblázatból, a helytelen azonosítások (hamis-pozitív és hamis-negatív) az egész (a finomítás előtti) adathalmaz 4,15%-át teszik ki. Ezekből néhány példa hírcím látható a 7.7. táblázatban.

7.7. táblázat

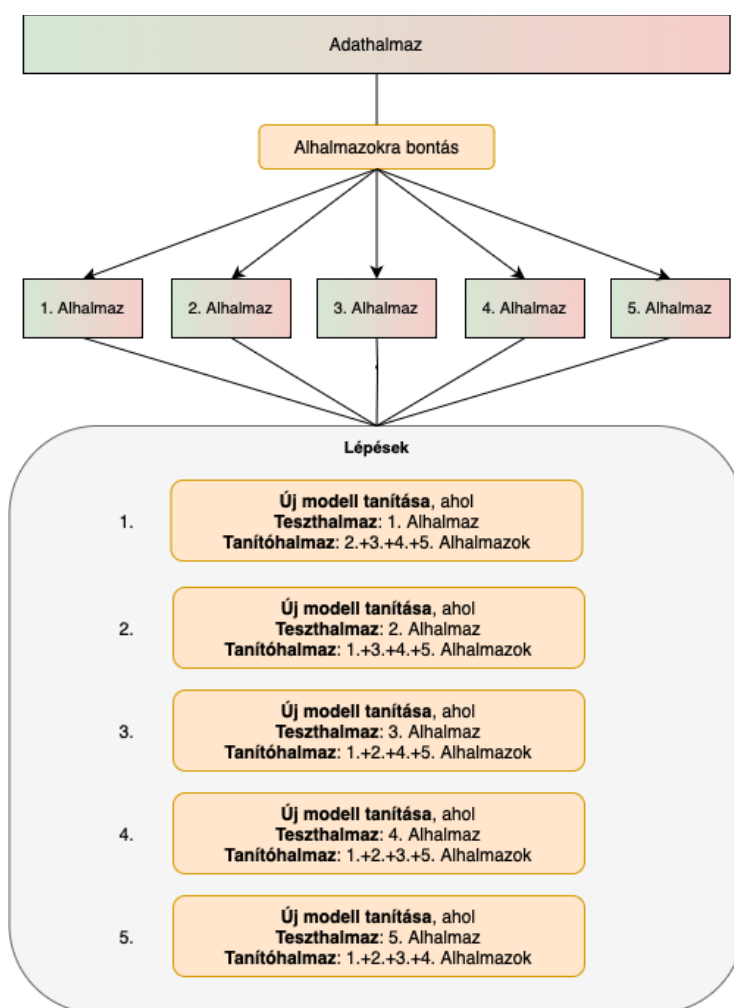
Hírcím	Hiba típusa
ELTŰNT EGY MAGYAR FALU PÉNZE, A POLGÁRMESTER KÜLFÖLDRE TÁVOZHATOTT	hamis-negatív
<NÉV> VENDÉGLŐJÉBEN KISZÁMLÁZTAK 200 FORINTOS JÉGKOCKÁT IS	hamis-negatív
Egyre kevésbé érik meg a <STREAMING PLATFORM>-nak az egyszerre ledarálható sorozatok	hamis-pozitív
Másfél perc maradt, és 13 pont a hátrány? Nem probléma, megnyerjük!	hamis-pozitív

Mint a táblázatból látható, a hamis-negatív (egy álhírt hírnek osztályozott a modell) hírcímeknek néhány esetben nem is álhíreknek kéne lenniük, így gyakorlatilag a modell osztályozása helyesebb volt, mint az én osztályozásom, amit az adatszűrési folyamat során végeztem. A hamis-pozitív esetén pedig ugyancsak felléphet az adatszűréskor félreosztályozott adat lehetősége, de a fent látható hírcímek alapján elmondható, hogy a szóhasználat és a mondat végi írásjelek zavarták meg a modellt az osztályozásban.

7.4. Utótesztelés

Hogy megbizonyosodjak a modell pontosságáról, a teljes adathalmazon végeztem egy keresztvalidációs tanítást és tesztelést. Fontos megemlíteni, hogy a finomítási fázis során gyűjtött adathalmazt is felhasználtam ehhez a tesztelési módszerhez, tehát összesen 13 111 hírcím állt a modell rendelkezésére, ebből 6498 álhír és 6613 hír kategóriájú.

Az utótesztelés folyamata hasonlóan zajlott a 7.2. fejezetben ismertetett módszerhez. Először kiválasztásra került egy az öt alhalmaz közül, ezt a kiválasztott alhalmazt teszhalmaznak, a többi alhalmazt pedig tanító halmaznak (továbbiakban: tanítóhalmazok) nevezem ki. Miután ez a kiválasztás lefutott, létrehozok egy tanítatlan modellt a 6.4. fejezet szerint már megállapított struktúrával és hiperparaméterekkel. Ezt a modellt aztán a tanítóhalmazokkal tanítom, ahol a tanítóhalmazokat egy halmazzá fűzöm össze a tanítás erejéig. Ezek után a kiválasztott teszhalmazon tesztelem a tanított modellt. Ezt megcsinálom annyiszor, ahány alhalmazom van, és minden lépés során egy olyan alhalmazt választok ki teszhalmaznak, ami még nem volt az. Az utótesztelési folyamatot ismerteti a 7.5. ábra.



7.5. ábra

A tesztek eredményeit a 7.8. táblázat ismerteti.

7.8. táblázat

Teszthalmaznak kiválasztott alhalmaz sorszáma	Teszthalmazon mért pontosság
1.	~95,49%
2.	~96,12%
3.	~96,00%
4.	~95,25%
5.	~95,49%

A modell átlag pontossága az utótesztelés során mért adatok alapján: ~95,67% [85]⁴.

Az ebben a tesztelésben használt tanítási módszer a teljes adathalmazt felhasználja, és nincs finomítási fázis, mint a 6.3. fejezetben, így a dolgozat során kifejlesztett modell valós pontosságát, továbbra is a 6.4. fejezetben megállapítottak fogom tekinteni.

⁴ Eltéréseket tartalmazhat a dolgozatban szereplő adatoktól (mint az adathalmaz mérete, tanítóciklusok száma stb.), ennek oka az, hogy a jegyzetfüzet a legfrissebb eredményeket tartalmazza, így ezt a dolgozat leadása után is folyamatosan frissítem.

8. Alkalmazhatóság és további feladatok

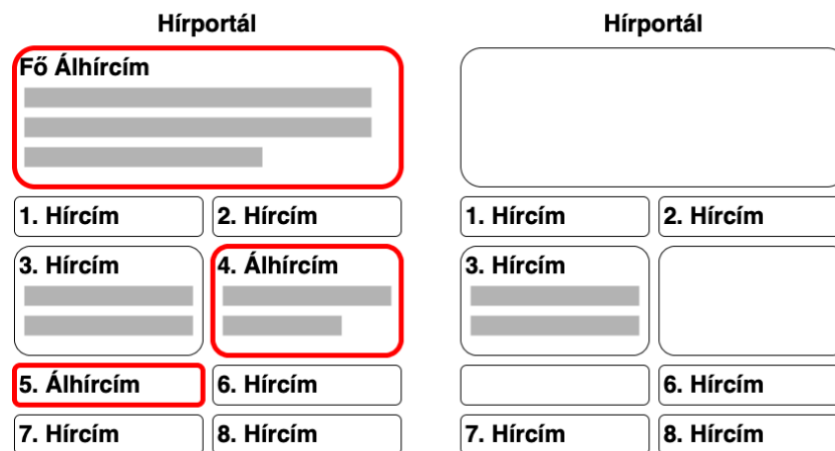
Ebben a fejezetben szeretném ismertetni kutatásom alkalmazhatóságát, és a további feladatokat, amik a kutatásban szereplő modell pontosságát vagy felhasználhatóságát is növelhetik.

8.1. Alkalmazhatóság

Ebben az alfejezetben ismertetem, hogy a kutatás eredményei milyen egyéb kutatásokra, vagy használati formákra adnak alapot. Mivel a betanított modell háttértárra van mentve, így bármilyen feladathoz felhasználható, vagy akár tovább is tanítható a későbbiekben.

8.1.1. Böngésző bővítmény

Az álhírek manapság az online térben érik el leginkább az embereket. Ha lenne egy olyan böngésző bővítmény, ami a kutatásban kifejlesztett modellt használja, akkor az képes lehetne figyelmeztetni az Internetet böngésző személyt az álhír jelenlétéről, akár az álhír kiemelésével vagy az álhír teljes eltüntetésével az éppen megnyitott weboldalról.



8.1. ábra

A bővítmény működésére látható egy szemléltetés a 8.1. ábrán. Mint ahogy az ábra is mutatja a bal oldalon, lehetséges egy olyan verzió, ahol az álhírnek hitt hírcímek csak kiemelve vannak a weboldalon (pirossal keretezettek), viszont ahogy a jobb oldalon is látszik, akár az álhírek eltüntetése is lehet egy megoldás.

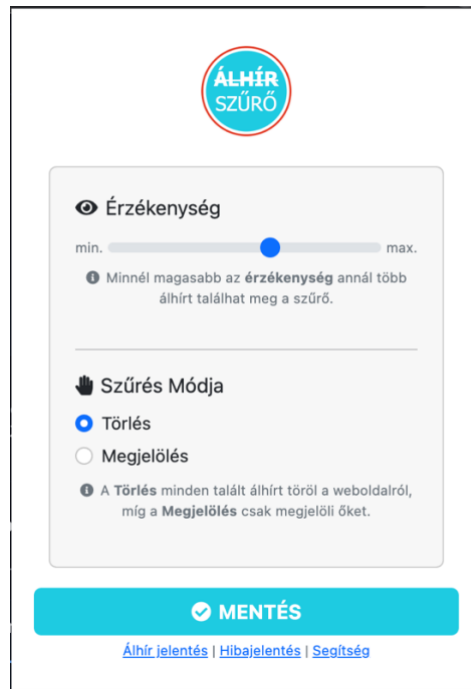
8.1.1.1. Megvalósított bővítmény

A modell kifejlesztése és finomítása után elkészítettem egy, az előzőekben leírt működésű bővítményt. Ennek létrejöttéhez szükséges volt egy szoftverrendszert fejlesztenem, amely egy webszerverből és egy böngésző bővítményből áll.

A webszerver feladata az, hogy egy adott végponton képes legyen a POST lekéréssel kapott bemeneti hírcímekhez álhír valószínűségét rendelni, ezt pedig válaszként visszaküldeni. Ehhez szükség volt arra, hogy a modell folyamatosan RAM-ban legyen a webszerveren, és hogy a szerver csak akkor válaszoljon egy kérésre, ha az a megfelelő hozzáférési kulcsot is bemenetként adja a lekérésben. A megvalósításához a Python programozási nyelvet és a Flask [74] szoftver könyvtárat használtam fel.

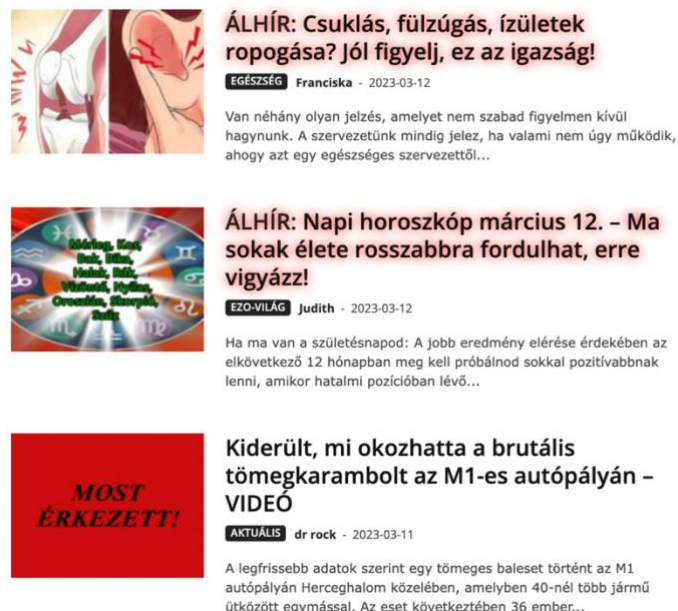
A böngésző bővítménynek több fontosabb feladata is van, ezeket az alábbi listában ismertetem:

- Megtalálni, illetve elmenteni az összes hírcímet és azok HTML elemeit, amit az éppen megnyitott weboldalon talál a bővítmény.
- Az elmentett hírcímeket előszűrni, azaz a HTML elemekből a tiszta szöveget kinyerni, és ha a szöveg túl rövid (kevesebb mint 8 karakter) vagy túl hosszú (több mint 1025 karakter), akkor azt a hírcímet elveti.
- Az előszűrt hírcímeket a megadott webszerver felé küldeni egy POST lekéréssel, ami tartalmazza a hozzáférési kulcsot.
- A webszerver válaszát feldolgozni, és a szerint törölni vagy megjelölni az éppen megnyitott weboldalon az álhíreknek vélt hírcímeket.
- A felugró ablak felületén keresztül a beállításokat kezelni, és azok beállított értékeit lementeni, hogy a böngésző újra nyitása során is ugyan azok az értékek legyenek használva.



8.2. ábra: A bővítmény felugró ablaka a Google Chrome böngészőben, természetesen a felhasználói felület tervezését, és a logót is saját magam terveztem és készítettem el

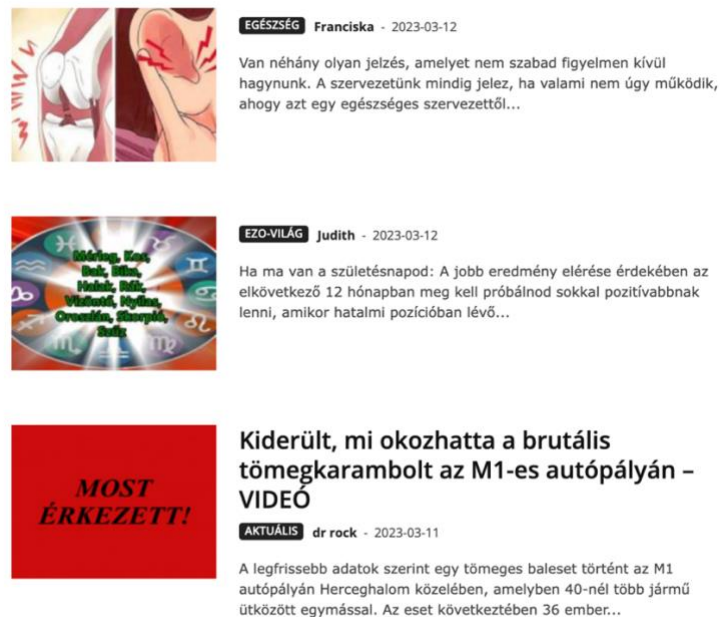
A 8.2. ábra szemlélteti a bővítmény felületét, ahol látható, hogy a bővítményben lehetőség van arra, hogy beállíthassuk az érzékenységet (kifejtve a 8.1.2. fejezetben), és a szűrés módját, ami lehet megjelölés (8.3. ábra) és törlés (8.4. ábra).



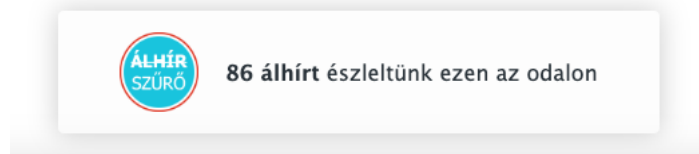
8.3. ábra: Mint látható, a megjelölés bizonyos szinten figyelemfelkeltőbbé teszi az álhíreket, ezért van lehetőség törlésre állítani a bővítményt, ami egyébként az alapértelmezett beállítás [tudasfaja.com]

A pontosabb szemléltetés érdekében, a képernyőképek készítésekor egy a feketelistáról származó weboldal volt megnyitva, 2023.03.12-én. Mint látható, a megjelölés egy „ÁLHÍR” feliratot helyez az adott hamisnak érzékelt hírcím elé, és kiemeli azt piros árnyékkal. Ha a

szűrési módszer törlésre van beállítva, akkor a bővítmény törli a hírcímhez tartozó HTML elemet.



8.4. ábra: A 8.3. ábrán is látható weboldal [tudasfaja.com] megnyitva a törlés beállítása után. Ha a bővítmény talál legalább egy álhírt a megnyitott weboldalon, akkor a felhasználót egy a weboldal alján, középen megjelenő értesítéssel értesíti (8.5. ábra). Ez az értesítés tartalmazza az oldalon talált álhírek számát, és 10 másodperc után láthatatlanná válik.



8.5. ábra

A bővítményt letölthetővé tettem [78 vagy 81], egy leírással együtt, ami tartalmazza, hogy hogyan kell telepíteni a Chrome böngészőben való használathoz [79 vagy 75].

8.1.2. Tűréshatár

A kutatás során azt állapítottam meg, hogy a modell akkor tekint egy hírcímet álhírnak, ha arra több mint 50%-os valószínűségi értéket ad kimenetként. Ez az érték, mivel egy a modelltől függetlenül osztályozó algoritmusba van beleírva, így lehetséges az, hogy az álhírek érzékelésének határát megváltoztassuk, azaz egy úgynevezett tűréshatárt állítunk be a modellel működő szoftveren.

Példa lehet erre a fentebb említett böngésző bővítmény. Ha egy felhasználó úgy találja, hogy a bővítmény túl sok hírcímet távolít el az általa böngészett weboldalokról, akkor a

tűrészatárt magasabbra is állíthatja (például 60%-ra) amivel a bővítmény már kevesebb hírt törölne ki a böngészés során. Matematikailag, ha $\hat{y} \in [0; 1]$ valós szám a modell kimenete, akkor a hírcím álhírnek számít, ha $\hat{y} \geq \omega$, ahol ω egy nulla és egy közötti valós szám, azaz a tűrészatár.

8.1.3. Automatikus ellenőrzés

A modellt akár írt szöveg ellenőrzésére is lehetne használni. Ebben az esetben például, ha valaki egy olyan oldalt üzemeltet, ahol a felhasználók szöveges tartalmakat oszthatnak meg nyilvánosan, akkor, ha a kutatás során elkészített modell ellenőrizné a szöveget a feltöltése előtt, hogy a tartalma mekkora valószínűséggel álhír, akkor akár ezzel a modell megelőzhetné az álhírek nyilvánosságra hozását azon az adott oldalon, ahol ellenőrzésre használták. A modell szövegek ellenőrzésére hasznos lehet, például: közösségi hálózatokon, blogmotor oldalakon, online áruházakban a termékértékeléseknél, csaló üzenetek szűréséhez a telefonokon, és még sok más esetben is.

8.1.3.1. Megvalósított kódtér

A Hugging Face oldalán, egy kódtér segítségével elérhetővé tettem a modellt kipróbálásra [82 vagy 76]. Ezen az oldalon egy szövegmező segítségével bármilyen bemenetet adhatunk a modellnek, és leellenőrizhetjük, hogy milyen álhír valószínűséget rendel hozzá.

8.2. További feladatok

Ebben a fejezetben ismertetem, hogy milyen feladatok vannak még vissza, amik növelhetik a dolgozatban bemutatott kutatás értékét, mint például a modell bővítése, vagy az adatbázissal való munkák.

8.2.1. További adatgyűjtés

Mint ahogy a modell finomításáról szóló fejezetben, itt is megemlítem, hogy minél több minőségi adat áll a modell rendelkezésére, annál pontosabb. Ennél fogva egy további feladat lehet az, hogy még több hírcímet gyűjtssek be a már meglévő hírcímek közé. Miután a gyűjtési folyamat lezárult és egy előre meghatározott számú hírcím összegyűlt, azokat a már említett szűrési módszerekkel szűrni kell, hogy a modellt ne tévesszék meg az esetlegesen a tanítóhalmazba kerülő anomáliák. Ezen kívül figyelni kell arra is, hogy már az adathalmazban lévő hírcímek csak egyszer szerepeljenek benne, tehát a tárolásnak redundancia mentesnek kell lennie.

8.2.2. Hírszövegek felhasználása

Még egy kiegészítése lehet a kutatásomnak az, ha a kifejlesztett modell a hírcímeken kívül a hírek szövegeit is felhasználja a tanuláshoz. Ezáltal egy pontosabb eredmény születhet, hiszen mint ahogy a tesztelésről szóló fejezetben bemutattam, a modellt a hírcímekben található karakterek variálása megtévesztheti, és valós híreket álhíreknek tekinthet (hamis-pozitív eset). Ám ha a teljes hírszöveg a modell rendelkezésére áll, akkor lehetséges, hogy nem csak a szó- vagy írásjel- használat, hanem a mondatok jelentése alapján is képes lesz megkülönböztetni az álhíreket a hírektől.

Ahhoz, hogy ez megvalósuljon, egy új modellt kell betanítani az új adatokon és az is lehetséges, hogy mind a modell struktúráját, és mind az adatfeldolgozási folyamatot is át kell alakítani a megváltozott bemenetek miatt.

8.2.3. Képek felhasználása

Ha a hírszövegek integrálása az adathalmazba és a tanítási folyamatba sikerrel zárulna, akkor lehetőség lehetne arra is, hogy a hírekhez tartozó képeket is felhasználjam a modell tanításához. Mivel nem minden hírhez tartozik egy kép, ezért ezt úgy lehetne a legkönnyebben megvalósítani, ha egy külön mély neurális hálózatot fejlesztenék arra a célra, hogy állapítsa meg képekről, hogy az álhírhez vagy valós hírhez való-e. Ezáltal, ha egy hírben szerepel egy vagy több kép, akkor az egyik modell a szöveg alapján, egy másik modell pedig a kép alapján határozza meg annak valószínűségét, hogy az adott hír álhír vagy sem. Ennek a két modellnek az eredményét aztán valamilyen arányszám szerint felhasználhatnám arra, hogy meghatározzák az adott hír kategóriáját. Például, ha $\hat{p} \in [0; 1]$ a képekkel dolgozó modell valószínűségi értéke és $\hat{t} \in [0; 1]$ a szöveggel működőé, akkor mondhatjuk, hogy egy hír álhír, ha $\frac{1}{5}\hat{p} + \frac{4}{5}\hat{t} \geq \omega$, ahol ω tűrészatár mértéke.

8.2.4. Időtállóság

Az éppen aktuális hírek és álhírek napról napra változhatnak, így a modellt érdemes folyamatosan tanítani az újonnan megjelenő adatokon. Például, lehet arról szó, hogy egy a modell adathalmazában eddig még nem szereplő összeesküvés elmélet lesz elterjedt az álhírek világában, és ebben az esetben nem biztos, hogy erről az új elméletről szóló hírcímekben képes felismerni a modell az álhírekre jellemző tulajdonságok alapján azt, hogy az a hír egy álhír-e. Így a modell tovább tanítása az új adatokon szükséges lehet.

9. Összefoglalás

Dolgozatom fő célkitűzése az volt, hogy kifejlesszek egy mély neurális hálózatot, ami a magyar nyelven írt álhírek beazonosítására specializálódott. A cél megvalósításához több lépés vezetett.

Első feladatomban a mesterséges intelligencia, a gépi tanulás és a mély tanulás alapos tanulmányozása volt, ehhez felhasználtam az ajánlott irodalom fejezetben megjelölt könyveket is. Ezek után a szakirodalmi áttekintés következett, melynek az volt a szerepe, hogy megismerjem a kutatásom főbb témaköreire kapcsolódó legfontosabb kutatásokat, és eredményeiket. Miután meghatároztam, hogy kutatásom létjogosultsága megalapozott, elkezdtem a felkészülést, melynek során első sorban adatokat kellett gyűjtenem különböző módszerekkel. Az adatokat szűrtem, és később a szakirodalmi áttekintés fázisában tanultak alapján kidolgoztam egy módszert, amivel az adathalmazban található magyar hírcímeket, a mély neurális hálózat számára érthető számszerű adatokká alakítottam. Ezek után elkezdtem a mély tanulási modell fejlesztését, ezzel párhuzamosan pedig az azt követő fejezetben található tesztelését is. A modell fejlesztése során végzett tesztekéről és módosításokról beszámolót készítettem, amit bemutattam a dolgozatom során. Az utolsó, teszteléssel kapcsolatos fejezetben pedig áthatóbban bemutattam a modell tesztelésének különböző módszereit.

A tesztalmez módszerrel mérve, a finomítási folyamattal tanított modell 97%-os pontosságot ért el (6.4. fejezet), míg a keresztvalidációs utóteszteléssel (7.4. fejezet) 95% lett a mért pontosság. Fontos megjegyezni, hogy a 95%-os modellt a teljes adathalmazon tanítottam és nem végeztem utólagos tanítást egy másik adathalmazzal.

A kutatás alkalmazhatóságának bemutatása érdekében, létrehoztam egy böngésző bővítményt (8.1.1.1. fejezet), ami valós időben szűri a megtekintett webhelyről az álhíreket, ezzel megvalósítva a bevezetőben említett célkitűzést is. Ez a bővítmény letölthető [78 vagy 81], és készítettem hozzá egy telepítési útmutatót is [79 vagy 75]. A bővítményen kívül, létrehoztam egy a Hugging Face oldalon elérhető kódteret (8.1.3.1. fejezet), aminek segítségével kipróbálható, hogy egy bemeneti hírcímre mekkora valószínűséget állapít meg a modell arra, hogy az egy álhír [82 vagy 76].

Az általam megvalósított modellről a kutatás során végzett tesztek és folyamatok alapján elmondható, hogy a célkitűzést teljesíti, és az álhíreket 97%-os pontossággal felismeri.

10. Magyarításjegyzék

10.1. táblázat: Az eredeti szavak és a dolgozatban használt magyarított megfelelőik

Dolgozatban használt magyarítások	Eredeti szavak
pontosvesszővel tagolt	csv (comma separated values)
kódtár	repository
teljesen kapcsolt	dense
konvolúciós	convolutional
hosszú rövid-távú memória	long short-term memory (LSTM)
konvolúciós neurális hálózatok	convolutional neural network (CNN)
visszacsatolt	recurrent
stopszavak	stopwords
szótövezés	stemming
adatfeldolgozás	tokenization
köteg méret	batch size
kiesési ráta	dropout rate
tanuló ciklusok száma	epochs
korai leállás	early stopping
túlillesztés	overfitting
átlag összevonó	average pooling
maximum összevonó	maximum pooling
k-szoros keresztvalidáció	k-fold cross-validation
simító	flatten
figyelem maszk	attention mask
zavarmátrix	confusion matrix
beágyazási	embedding
összevonó	pooling
réteg	layer
visszatérés	return
transzfer tanulás	transfer learning

11. Ábrajegyzék

2.1. ábra: A mély tanulás elhelyezkedése a mesterséges intelligencia témakörében. [leet.hu]	3
2.2. ábra: bal oldalon az elmozdulásvektor (lábnymokkal), jobb oldalon a rugalmas testen alakváltozást okozó erő vektora.....	4
3.1. ábra: A mondat szintű feldolgozás szemléltetése példa mondattal	9
3.2. ábra: Példa a szavak szintjén történő feldolgozásra, a 3.1 ábrán látható szöveget tovább használva	9
4.1. ábra: Képernyőkép a Hugging Face oldalán kipróbálható modellről [82 vagy 76]	12
4.2. ábra: Egy böngészőből megnyitott Jupyter Notebook jegyzetfüzet a Google Colab oldalán	12
5.1. ábra: A Firefox webböngésző távvezérlés módban megnyitva a 24.hu oldalán [24.hu]	16
5.2. ábra: Az adathalmaz két osztályának eloszlása százalékosan	20
5.3. ábra: A hírek szótérképe.....	20
5.4. ábra: Az álhírek szótérképe.....	21
5.5. ábra: Az adathalmazokban található 10 leggyakoribb szó (mind a két kategória egyben)	21
5.6. ábra: Szemlélteti, hogy szavakra bontás után, minden szóhoz egy számot egy rendelünk (adatfeldolgozás) 23	
5.7. ábra: Az alapnak vett „Többmillió tömeg gyűlt össze Buenos Airesben!” mondatra alkalmaztam többfajta szövegfeldolgozási módszert, aminek az eredményei láthatóak a bal oldalon, az ábrán látható, hogy nem mindegy, hogy egy adott szöveget milyen módon dolgozok fel, hiszen teljesen más számok jöhetnek ki kimenetként az adatfeldolgozó algoritmusból (amik az ábrán a jobb oldalon vannak szemléltetve).....	24
5.8. ábra: Példa egy szókincstáblázatra, ha egy szót nem találunk a táblázatban, akkor a <NULL> szó számát fogja kapni, a „...” a JSON objektum végén jelöli, hogy a táblázat még más szavakat is tartalmaz az ábrán láthatóak mellett	24
5.9. ábra: Mint látható a „többmillió”, „buenos” és az „airesben” szavak nem voltak benne a szókincstáblázatban, így a <NULL> értékét kapták, ami 1	25
5.10. ábra: Folyamatábra az alapnak vett algoritmus működéséről, mellette pedig egy példa hírcím feldolgozása látható lépésről lépésre	27
5.11. ábra: Az ábrán látható a referenciaként használt teljesen kapcsolt rétegekkel ellátott neurális hálózat struktúrája, a réteg reprezentációján látható az aktivációs függvény neve, vonallal odakötve pedig a réteg típusa és a kimenetének dimenziói	28
5.12. ábra: A szövegfeldolgozó módszer bemutatása, jobb oldalt egy példa hírcímmel.....	33
5.13. ábra: Az adatfeldolgozás metódusai, még a fejlesztési folyamat során, a fent látható metódus kicseréli az ékezetes betűket, az alsó metódus pedig a teljes szövegfeldolgozási folyamatot végrehajtja egy paraméterként kapott mondaton („text” paraméter), mind a két metódus Reguláris Kifejezések használatát tartalmazza.....	33
6.1. ábra: Az alapnak vett mély neurális hálózat.....	36
6.2. ábra.....	38
6.3. ábra.....	39
6.4. ábra: Látható, hogy a bemeneti réteg most már minden bemenethez kap egy figyelem maszkot is.....	41
6.5. ábra.....	42
6.6. ábra.....	47
6.7. ábra: A modell inicializálásához írt forráskód, a kép még a fejlesztési folyamat során készült.....	48
6.8. ábra: A modell tanításához írt forráskód, ezen látható a hibafüggvény, az optimalizáló és a korai leállás is	48
7.1. ábra.....	50
7.2. ábra.....	52

7.3. ábra.....	53
7.4. ábra.....	56
7.5. ábra.....	57
8.1. ábra.....	59
8.2. ábra: A bővítmény felugró ablaka a Google Chrome böngészőben, természetesen a felhasználói felület tervezését, és a logót is saját magam terveztem és készítettem el.....	61
8.3. ábra: Mint látható, a megjelölés bizonyos szinten figyelemfelkeltőbbé teszi az álhíreket, ezért van lehetőség törlésre állítani a bővítményt, ami egyébként az alapértelmezett beállítás [tudasfaja.com]	61
8.4. ábra: A 8.3. ábrán is látható weboldal [tudasfaja.com] megnyitva a törlés beállítása után	62
8.5. ábra.....	62

12. Táblázatjegyzék

3.1. táblázat	7
4.1. táblázat	14
5.1. táblázat	18
5.2. táblázat	22
5.3. táblázat	22
5.4. táblázat	29
5.5. táblázat	29
5.6. táblázat	30
5.7. táblázat	30
5.8. táblázat	31
5.9. táblázat	32
5.10. táblázat	32
6.1. táblázat	36
6.2. táblázat	37
6.3. táblázat	38
6.4. táblázat	39
6.5. táblázat	40
6.6. táblázat	40
6.7. táblázat	41
6.8. táblázat	42
6.9. táblázat	43
6.10. táblázat	43
6.11. táblázat	46
6.12. táblázat	46
6.13. táblázat	48
6.14. táblázat	49
6.15. táblázat	49
7.1. táblázat	53
7.2. táblázat	53
7.3. táblázat	54
7.4. táblázat	54
7.5. táblázat	55
7.6. táblázat	55
7.7. táblázat	56
7.8. táblázat	58
10.1. táblázat: Az eredeti szavak és a dolgozatban használt magyarított megfelelőik.....	66

13. Ajánlott irodalom

- Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016
- Wettl, F. (2011). Lineáris Algebra. *Typotex, Budapest.*
- István, F. (2013). Neurális hálózatok.
- György, C. (2020). *Ember 2.0: A mesterséges intelligencia gazdasági és társadalmi hatásai.* Kossuth Kiadó.
- Gyarmati, P. (2019). Gondolatok a mesterséges intelligencia, gépi tanulás kapcsán. *Mesterséges Intelligencia, 1(1), 31-39.*
- IE, B. Mély konvolúciós neurális hálózatok.
- Krekó, P., & Falyuna, N. (2021). Sarlatánok kora – Miért dőlünk be az áltudományoknak?. *Athenaeum Kiadó.*
- Pataki B. Á.. (2019). Deep learning alapok. *ELTE.*

14. Irodalomjegyzék

- [1] Jiang, T. A. O., Li, J. P., Haq, A. U., Saboor, A., & Ali, A. (2021). A novel stacking approach for accurate detection of fake news. *IEEE Access*, 9, 22626-22639.
- [2] Veronika, V. MSZNY 2018.
- [3] Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2021). Deep learning--based text classification: a comprehensive review. *ACM Computing Surveys (CSUR)*, 54(3), 1-40.
- [4] Márk, N. D. (2022). Introducing huBERT. XVII. Magyar Számítógépes Nyelvészeti Konferencia, 3-14.
- [5] Gencsi, M., Bodó, Z., & Szenkovits, A. (2021, September). Compilation and Validation of a Large Fake News Dataset in Hungarian. In *2021 IEEE 19th International Symposium on Intelligent Systems and Informatics (SISY)* (pp. 125-130). IEEE.
- [6] Wettl, F. (2011). *Lineáris Algebra. Typotex, Budapest.*
- [7] Kok, J. N., Boers, E. J., Kusters, W. A., Van der Putten, P., & Poel, M. (2009). Artificial intelligence: definition, trends, techniques, and cases. *Artificial intelligence*, 1, 270-299.
- [8] Mitchell, T. M., & Mitchell, T. M. (1997). *Machine learning* (Vol. 1, No. 9). New York: McGraw-hill.
- [9] Hwang, E. J., Park, S., Jin, K. N., Im Kim, J., Choi, S. Y., Lee, J. H., ... & Park, C. M. (2019). Development and validation of a deep learning--based automated detection algorithm for major thoracic diseases on chest radiographs. *JAMA network open*, 2(3), e191095-e191095.
- [10] Ardila, D., Kiraly, A. P., Bharadwaj, S., Choi, B., Reicher, J. J., Peng, L., ... & Shetty, S. (2019). End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature medicine*, 25(6), 954-961.
- [11] Deng, L., & Yu, D. (2014). Deep learning: methods and applications. *Foundations and trends® in signal processing*, 7(3-4), 197-387.
- [12] Li, P., Rao, X., Blase, J., Zhang, Y., Chu, X., & Zhang, C. (2021, April). CleanML: a study for evaluating the impact of data cleaning on ML classification tasks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)* (pp. 13-24). IEEE.
- [13] Ágnes, V. (2017). Az álhírek extra-és intralingvális jellemzői. *Századvég*, 84(2), 51-84.
- [14] Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.
- [15] Chacon, S., & Straub, B. (2014). *Pro git*. Apress.
- [16] GitHub. (2020). *GitHub*. <https://github.com/>.
- [17] Hugging Face. (2022). *Hugging Face*. <https://huggingface.co/>.
- [18] Kaminska, I. (2017). A module in fake news from the info-wars of ancient Rome. *Financial Times*. Accessed, 28, 03-18.
- [19] Thota, A., Tilak, P., Ahluwalia, S., & Lohia, N. (2018). Fake news detection: a deep learning approach. *SMU Data Science Review*, 1(3), 10.
- [20] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
- [21] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265-283).
- [22] Gulli, A., & Pal, S. (2017). *Deep learning with Keras*. Packt Publishing Ltd.
- [23] Kramer, O. (2016). Scikit-learn. In *Machine learning for evolution strategies* (pp. 45-53). Springer, Cham.
- [24] Willett, P. (2006). The Porter stemming algorithm: then and now. *Program*.
- [25] Zhang, Z. (2018, June). Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)* (pp. 1-2). Ieee.

- [26] Ruby, U., & Yendapalli, V. (2020). Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(10).
- [27] Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.
- [28] Salazar, J. J., Garland, L., Ochoa, J., & Pyrcz, M. J. (2022). Fair train-test split in machine learning: Mitigating spatial autocorrelation for improved prediction accuracy. *Journal of Petroleum Science and Engineering*, 209, 109885.
- [29] Nemeskey, D. M. (2020). Natural language processing methods for language modeling.
- [30] Li, M., Zhang, T., Chen, Y., & Smola, A. J. (2014, August). Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 661-670).
- [31] Eelbode, T., Sinonquel, P., Maes, F., & Bisschops, R. (2021). Pitfalls in training and validation of deep learning systems. *Best Practice & Research Clinical Gastroenterology*, 52, 101712.
- [32] Prechelt, L. (1998). Early stopping-but when?. In *Neural Networks: Tricks of the trade* (pp. 55-69). Springer, Berlin, Heidelberg.
- [33] Máté, K. D. A GÉPI TANULÁS ALAPJAI. *Tavaszi Szél–Spring Wind 2019*, 92.
- [34] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.
- [35] Asaad, R. R., & Ali, R. I. (2019). Back Propagation Neural Network (BPNN) and sigmoid activation function in multi-layer networks. *Academic Journal of Nawroz University*, 8(4), 216-221.
- [36] Jouppi, N., Young, C., Patil, N., & Patterson, D. (2018). Motivation for and evaluation of the first tensor processing unit. *IEEE Micro*, 38(3), 10-19.
- [37] Steinkraus, D., Buck, I., & Simard, P. Y. (2005, August). Using GPUs for machine learning algorithms. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)* (pp. 1115-1120). IEEE.
- [38] Gholamalizadeh, H., & Khosravi, H. (2020). Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*.
- [39] Muhdalifah, M. F. (2022). Pooling comparison in CNN architecture for Javanese script classification. *International Journal of Informatics and Computation*, 3(2), 15-22.
- [40] Sharmila, A., EL, D. P., Anand, K. G., Sujin, J. S., Soundarya, B., & Krishnaraj, R. (2022, January). Fruit Recognition Approach by Incorporating Multilayer Convolution Neural Network. In *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)* (pp. 1042-1046). IEEE.
- [41] Malik, P., Aggrawal, A., & Vishwakarma, D. K. (2021, April). Toxic speech detection using traditional machine learning models and BERT and fasttext embedding with deep neural networks. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 1254-1259). IEEE.
- [42] Tenney, I., Das, D., & Pavlick, E. (2019). BERT rediscovers the classical NLP pipeline. *arXiv preprint arXiv:1905.05950*.
- [43] Wikipédia. (2020). *Wikipédia*. <https://hu.wikipedia.org/wiki/>.
- [44] Hrinchuk, O., Khrulkov, V., Mirvakhabova, L., Orlova, E., & Oseledets, I. (2020, November). Tensorized embedding layers. In *Findings of the Association for Computational Linguistics: EMNLP 2020* (pp. 4847-4860).
- [45] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929-1958.
- [46] Lawrence, S., Giles, C. L., & Tsoi, A. C. (1998). What size neural network gives optimal generalization? Convergence properties of backpropagation.
- [47] Torrey, L., & Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques* (pp. 242-264). IGI global.
- [48] Ács, J., Lévai, D., Nemeskey, D. M., & Kornai, A. (2021). Evaluating contextualized language models for hungarian. *arXiv preprint arXiv:2102.10848*.

-
- [49] Mitchell, R. (2018). Web scraping with Python: Collecting more data from the modern web. " O'Reilly Media, Inc."
 - [50] FERENC, P. V. Á., & MÁTÉ, F. HA CSAK EGY TANULMÁNYT OLVAS EL, EZ LEGYEN AZ!–AVAGY MI AZ A CLICKBAITING, ÉS MIÉRT UTÁLJA MINDENKI?. *UNIVERSITAS BUDAPESTIENSIS DE "METROPOLITAN"*, 124.
 - [51] Krekó, P. (2021). Tömegparanoia 2.0. Összeesküvés-elméletek, álhírek és dezinformáció. *Athenaeum Kiadó*.
 - [52] Krekó, P., & Falyuna, N. (2021). Sarlatánok kora – Miért dőlünk be az áltudományoknak?. *Athenaeum Kiadó*.
 - [53] Porter, M. F. (2001). Snowball: A language for stemming algorithms.
 - [54] Shapiro, L. G., & Stockman, G. C. (2001). *Computer vision* (Vol. 3). New Jersey: Prentice Hall.
 - [55] Jacovi, A., Shalom, O. S., & Goldberg, Y. (2018). Understanding convolutional neural networks for text classification. *arXiv preprint arXiv:1809.08037*.
 - [56] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436-444.
 - [57] Jin, J., Dunder, A., & Culurciello, E. (2014). Flattened convolutional neural networks for feedforward acceleration. *arXiv preprint arXiv:1412.5474*.
 - [58] Douglas, S. C., & Yu, J. (2018, October). Why RELU units sometimes die: analysis of single-unit error backpropagation in neural networks. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers* (pp. 864-868). IEEE.
 - [59] Dubey, A. K., & Jain, V. (2019). Comparative study of convolution neural network's relu and leaky-relu activation functions. In *Applications of Computing, Automation and Wireless Systems in Electrical Engineering* (pp. 873-880). Springer, Singapore.
 - [60] Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2013). How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.
 - [61] Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM--a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*.
 - [62] Bontemps, L., Cao, V. L., McDermott, J., & Le-Khac, N. A. (2016, November). Collective anomaly detection based on long short-term memory recurrent neural networks. In *International conference on future data and security engineering* (pp. 141-152). Springer, Cham.
 - [63] Irie, K., Tüske, Z., Alkhouli, T., Schlüter, R., & Ney, H. (2016, September). LSTM, GRU, highway and a bit of attention: an empirical overview for language modeling in speech recognition. In *Interspeech* (pp. 3519-3523).
 - [64] Hassan, A., & Mahmood, A. (2018). Convolutional recurrent deep learning model for sentence classification. *Ieee Access*, 6, 13949-13957.
 - [65] Sharma, N., & Saroha, K. (2015, May). Study of dimension reduction methodologies in data mining. In *International Conference on Computing, Communication & Automation* (pp. 133-137). IEEE.
 - [66] Noémi, L. N., Gergő, F., Enikő, H., Kinga, J. M., János, L. L., Noémi, V., ... & Tamás, V. HuLU: magyar nyelvű benchmark adatbázis kiépítése a neurális nyelvmodellek kiértékelése céljából.
 - [67] Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., & Ridella, S. (2012). The 'K' in K-fold cross validation. In *20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)* (pp. 441-446). i6doc. com publ.
 - [68] Microsoft Corporation. (2022). *Microsoft Excel*. <https://office.microsoft.com/excel>
 - [69] Cunningham, P., Cord, M., & Delany, S. J. (2008). Supervised learning. In *Machine learning techniques for multimedia* (pp. 21-49). Springer, Berlin, Heidelberg.
 - [70] Visa, S., Ramsay, B., Ralescu, A. L., & Van Der Knaap, E. (2011). Confusion matrix-based feature selection. *MAICS*, 710(1), 120-127.
 - [71] Diagrams.net. (2022). *Diagrams.net*. <https://app.diagrams.net>
-

- [72] Godbole, V., Dahl, G. E., Gilmer, J., Shallue, C. J., & Nado, Z. (2023). *Deep Learning Tuning Playbook*. https://github.com/google/tuning_playbook
- [73] Posetti, J., & Matthews, A. (2018). A short guide to the history of ‘fake news’ and disinformation. *International Center for Journalists*, 7(2018), 2018-07.
- [74] Grinberg, M. (2018). *Flask web development: developing web applications with python*. " O'Reilly Media, Inc."
- [75] Kondor, M. (2023). *Magyar nyelvű álhírek felismerése mély neurális hálózattal – Bővítmény telepítési útmutató* – Másodlagos link. <https://drive.google.com/file/d/1Cm-NQkLXfzF5yZHZeT5Ko1lJv4xsBhgD/view?usp=sharing>
- [76] Kondor, M. (2023). *Magyar nyelvű álhírek felismerése mély neurális hálózattal - Kódtér*. <https://huggingface.co/spaces/martinkondor/fake-news-detection-hu>
- [77] Kondor, M. (2023). *Magyar Álhír Felismerés - Álhír jelentő űrlap*. <https://forms.gle/AmxtpNz12NkSwa7T9>
- [78] Kondor, M. (2023). *Magyar nyelvű álhírek felismerése mély neurális hálózattal – Bővítmény*. <http://193.225.122.151/serve/bovitmeny.zip>
- [79] Kondor, M. (2023). *Magyar nyelvű álhírek felismerése mély neurális hálózattal – Bővítmény telepítési útmutató*. http://193.225.122.151/serve/bovitmeny_utmutato.pdf
- [80] NMHH. (2022). *Internetes közönségmérési adatok (2022 I. negyedév)*. https://nmhh.hu/cikk/228634/Internetes_kozonsegmeresi_adatok_2022_I_negyedev
- [81] Kondor, M. (2023). *Magyar nyelvű álhírek felismerése mély neurális hálózattal – Bővítmény – Másodlagos link*. <https://drive.google.com/file/d/1stoy9Yjwlv54mlBkWhVvTmhtPj7VmFq3/view?usp=sharing>
- [82] Kondor, M. (2023). *Magyar nyelvű álhírek felismerése mély neurális hálózattal – Kódtér – Másodlagos link*. <http://193.225.122.151/serve/index.html>
- [83] Kondor, M. (2023). *Magyar nyelvű álhírek felismerése mély neurális hálózattal – Jegyzetfüzet*. <http://193.225.122.151/serve/tanitas.ipynb>
- [84] Kondor, M. (2023). *Magyar nyelvű álhírek felismerése mély neurális hálózattal – Adathalmaz*. http://193.225.122.151/serve/all_data.csv
- [85] Kondor, M. (2023). *Magyar nyelvű álhírek felismerése mély neurális hálózattal – Utótesztelés*. <http://193.225.122.151/serve/utoteszt.ipynb>