

TDK dolgozat

Czetin Márk

Kovács Hunor György

2024

Hír elemző és kiértékelő gépi tanulás-alapú keretrendszer a piaci környezet leírására el- adási adatok előrejelzéséhez

Czetin Márk

Kovács Hunor György

Széchenyi István Egyetem

Gépészmérnöki, Informatikai és Villamosmérnöki Kar

Mérnökinformatikus szak, BSc 4. év

Témavezető:

Hajdu Csaba, egyetemi tanársegéd

dr. Bognár Tamás, egyetemi adjunktus

Kézirat lezárásának dátuma: 2024.11.04

Absztrakt

Hír elemző és kiértékelő gépi tanulás-alapú keretrendszer a piaci környezet leírására eladási adatok előrejelzéséhez

Az elmúlt években számos globális esemény történt, amely jelentős mértékben megváltoztatta a fogyasztói társadalom viselkedését. Ezáltal egy termék eladásainak előrejelzése szinte lehetlenné vagy rendkívül erőforrás-igényessé vált. Ez egyre nagyobb probléma a logisztika és az értékesítés területén, különösen a nagy vállalatok körében. További kihívást jelent, hogy gyakran hiányos, vagy nem létező az adatáramlás egy adott vállalat és az értékesítési csatornái között, így a sellout adatok hiánya tovább nehezíti a jövőbeli eladások becslését. Ezeket a nehézségeket a kutatásban együttműködő nagyvállalat is azonosította. A kutatás célja egy gépi tanulás alapú döntéstámogató rendszer megvalósítása. Ennek a rendszernek jobbnak, vagy legalább olyan jónak kell lennie, mint a vállalat szakértői által becsült előrejelzések, miközben jelentősen alacsonyabb az erőforrás igénye. A gépi tanulási modell egy Long-Short Term Memory (LSTM) alapú architektúrára épülő modell. Ehhez a titkosított adatokat a vállalat biztosította, melyeknek a részletes feldolgozása a kutatás korábbi részét képezi. Az eladások előrejelzéséhez nem elegendő kizárólag a múltbeli historikus adatokra támaszkodni, mivel ezek önmagukban nem adnak magyarázatot az adatok között fennálló korrelációra. Ennek a korlátozásnak leküzdésére különböző hírportálok hírein szentiment analízis lett alkalmazva egy Retrieval-Augmented Generation (RAG) rendszer segítségével, amely lehetővé teszi az ok-okozati összefüggések megfigyelését az eladások között, és ezáltal figyelembe venni ezeket az előrejelzések folyamán. A kutatás jelentős eredményeket ért el a megvalósíthatóság és az adatfeldolgozás területén. Jelenleg a keretrendszer képes az eladások előrejelzésére, a jövőbeni cél pedig egy olyan teljes döntéstámogató szolgáltatás kifejlesztése, amely képes alternatív stratégiák előállítására is különböző bemenő paraméterek esetén. Továbbá fontos, hogy ez a keretrendszer minden divízió számára elérhetővé váljon a jövőben.

Köszönetnyilvánítás

Hálával tartozunk felkészítő tanárainknak, Hajdu Csaba és dr. Bognár Tamás Tanár Úrnak aldozatos munkájukért, amit a dolgozat elkészülése során tanúsítottak. Észrevételeik, motiválásuk és szakmai tapasztalatuk nagyon sokat hoztátett mind tartalmi, mind stilisztikai szempontból a kutatás minőségéhez. Valamint szeretnénk megköszönni, a partner vállalatnak és külön köszönet a kapcsolattartó személynek, aki segített betekintést nyerni a vállalat működésébe. Köszönjük szépen a lehetőséget nekik és örömmel várjuk a további együttműködést a jövőben is.

Tartalomjegyzék

1.	Bevezetés.....	3
2.	A vállalati környezet	5
2.1	A vállalat felépítése	5
2.2	Agilis működés a vállalatban belül	6
2.3	Forecast a vállalatban belül.....	6
3.	Szakirodalmi áttekintés	10
3.1	Neurális hálózat alapú rendszerek.....	10
3.2	A választott architektúra ismertetése.....	12
3.2.1	A választott rekurrens modell	12
3.2.2	Az LSTM modell neuronjai	14
3.2.3	Az LSTM modell kapujai.....	14
3.2.4	Bidirectional LSTM	16
3.3	Természetes nyelvi feldolgozás.....	19
3.3.1	Szavak értelmezése gépi tanuláson keresztül.....	19
3.3.2	Nagy nyelvi modellek	21
3.3.3	Lemmatizáció és stemmelés.....	22
3.3.4	Széntiment analízis.....	23
3.3.5	RAG rendszer.....	24
4.	Prototípus szoftver.....	27
4.1	Directional LSTM előrejelzés nem vállalati adatokon.....	27
4.2	Bidirectional LSTM előrejelzés vállalati adatokon.....	27
5.	A hírfeldolgozáshoz fejlesztett szoftver	31
5.1	A hírfeldolgozás célja a kutatás folyamán	31
5.2	Adatgyűjtés a hírportálokról.....	33
5.3	A begyűjtött adatok feldolgozása, adatbázisba rendezése.....	37

5.4	Szentiment analízis LLM felhasználásával	41
5.5	RAG architektúrára épülő szentiment analízis	45
6.	Az eredmények.....	49
7.	Konklúzió.....	51
7.1	Jövőbeli munka	51
ÁBRAJEGYZÉK		54
TÁBLÁZATJEGYZÉK.....		56
IRODALOMJEGYZÉK		57

1. Bevezetés

A nagyvállalatok effektív működéséhez elengedhetetlen a megfelelő tervezés. Ez ad lehetőséget arra, hogy az ellátási lánc minden pontja zökkenőmentesen legyen képes együttműködni. Így a cégek pontos képet kapnak arról, hogy milyen eladásra számíthatnak egyes értékesítési csatornákon a konkrét termékeik esetén, amely a láncnak a legvége az értékesítésből származó bevétel beérkezése, ami így mindent befolyásol, ami felette van, ezáltal a gyártást, szállítást, a marketinget, az értékesítést és a pénzügyet, tehát minden lépést, ami tervezést igényel. A felsorolt folyamatok konstrukciójához szüksége van a cégnek olyan számszerű adatokra, amelyek alapján ők is számszerűen képesek megbecsülni az eladásokat, akár értékben, mennyiségben, vagy meg tudják állapítani a gyárba rendelendő csomagolóanyagot például. Ez a számszerű adat a forecast, amely egy előrejelzés, amit a cégben dolgozó személyek becsülnek meg piaci helyzet elemzéséből, termelési környezetből, rendelkezésre álló tudásukkal és előző éves adatokkal. Ez alapvetően is helyet ad az emberi hibáknak, mint a pontatlanság, fáradtság, érzelmi befolyásolt-ság vagy a szubjektív vélemény túlsúlyossága. Ezen kívül, a becslés folyamatában jelentős szerepet játszik a dolgozók tapasztalata. Azonban az elmúlt pár év szélsőséges piaci helyzeteire nem alkalmazható sem az előtte lévő években gyűjtött adat, sem tapasztalat, hiszen az igények és a trendek kritikusan megváltoztak kereskedelemben az elmúlt idők globális eseményei, valamint az amúgy is egyre hektikusabb piaci dinamika hatására. Ahhoz, hogy a cég újra megfelelően tudjon tervezni az aktuális piaci igényeknek megfelelően, agilisabban kell működnie. Ez azt jelenti, hogy nem feltétlen tudja a vállalat tartani magát ahhoz a forecasthoz, amit egy éve becsültek meg, majd esetleg minimálisan módosítottak időközben, hogyha az év folyamán jelentősen megváltoztak a piaci körülmények. Erre viszont jelenleg nem képesek, mert egy forecast megállapítása jelentős időbe és erőforrásba kerül, így, ha az idő múlásával kiderül, hogy új tervet kell készíteniük és követniük, már nem tehetik meg, mert túl sok idő és erőforrás lenne azt kialakítani majd átállni rá.

A kutatás célja, hogy olyan döntéstámogató rendszer (DTR) készüljön el általa, ami megoldja ezt a problémát. Tehát, a cég legyen képes agilisán forecastot előállítani, amely lehetőséget ad nekik új, alternatív stratégiákat kialakítani anélkül, hogy ebbe újabb jelentős időt, illetve emberi erőforrást fektetnének. A dolgozat folyamán ennek a rendszernek a vállalatirányításon belüli támogatása és leképezése történik meg. A szoftver elkészüléséhez különböző gépi tanulós módszerek kerültek felhasználásra.

A rendszer a forecast előállítását két irányból közelíti meg, hasonlóan ahhoz, ahogy a szakértők a vállalatnál is tennék ezt.

Elsődlegesen, a múltbeli adatokat felhasználva egy Long Short-Term Memory (LSTM) architektúrára épülő modell került kialakításra, amely lehetővé tette, hogy az adatokat szekvenciális sorrendben dolgozza fel a rendszer, így az képes volt tanulni az információk időbeliségéből is. Ezáltal lehetséges az előző éves eladásokból különböző trendeket és fogyasztói szokásokat megfigyelni. Ehhez a tanításhoz nagy mennyiségű adatra van szükség az előző évekből, amelyeket a cég szolgáltat a kutatás folyamán. Ezek az adatok sok helyen hiányosak, vagy modell tanításra nem megfelelő formájúak voltak, így a kapott adatok jelentős feldolgozásra kerültek az LSTM modellhez.

A rendszer másodlagosan a valós idejű híreket dolgozza fel a megfelelő előrejelzésekhez. Ehhez az internetről gyűjt Application Programming Interface-n (API) keresztül híreket hírportálokról és központi adattárakból. Ezután Natural Language Processing (NLP) segítségével ezeket a híreket feldolgozza és kiértékeli egy Large Language Model (LLM). Ez az LLM egy Retrieval Augmented Generation (RAG) rendszer részét képezi. A RAG rendszer képes nagy mennyiségű írásos információ tárolására és felhasználására egy nyelvi feldolgozás folyamán, amire egy LLM önmagában nem képes. A kiértékelés eredménye több előre meghatározott paraméter számosított értéke, mint például, hogy az adott hír hordoz e magában rizikót a cég szempontjából, vagy sem.

A rendszer az LSTM és a RAG segítségével képes mind a több évre visszanyúló múltbeli adatok mind pedig az elmúlt pár hetes, de akár valós idejű adatokat felhasználva is előre jelezni a jövőbeli eladásokat. Ez a rendszer lehetőséget ad a cégnek, hogy gyorsan, kellően pontos előrejelzéseket állítsanak elő sokkal kevesebb idő alatt, mint amilyeneket emberi úton képesek lennének. Így lehetősége nyílik a vállalatnak az agilis működésre, ami által újra naprakészek tudnak maradni, és megfelelő stratégiai tervet tudnak követni

2. A vállalati környezet

2.1 A vállalat felépítése

A kutatás által érintett vállalat üzletág alapon működik, ami azt jelenti, hogy különböző divíziókra osztható. Összesen 7 ilyen fő divízió létezik, melyek a következők:

- Csokoládé készítmények
- Konyhai készítmények
- Bébiételek
- Állateledek
- Kávé készítmények
- Fizetős termék automaták
- Gabonapehely készítmények

A kutatás folyamán fejlesztett szoftver jelenleg a magyarországi gabonapehely divízióval (MGD) foglalkozik. A gabonapehely divízió (GD) egy vegyes vállalat egy vállalatcsoporton belül, amely egy amerikai és egy európai vállalat együttműködéséből jött létre. Ebben konszernben a termékeket az amerikai cég, míg az arculatot az európai adta. Ezáltal megkülönböztetünk egy külsős és egy belsős együttműködést. A külsős együttműködés egy Európa szintű hálózat, amelynek tagja Magyarország, Csehország, Lengyelország és Szlovákia. A belsős együttműködés pedig a Magyarországon belül működő divíziók hálózata.

Az európai standardot a külsős együttműködés, tehát a közép európai csoport adja a tevékenységekre és az arculatra vonatkozóan. Ebben a klaszterban a termékek bevezetése, tervezése, dizájnolása együttes megegyezés alapján történik. Ez azért nehezíti meg a tervezési folyamatokat, mert a fogyasztói szokások jelentősen eltérőek Lengyelországban, mint például Magyarországon. Így egy termék bevezetése például nagyon nehéz feladat, hiszen szűk az a keresztmetszet, amiben minden ország igényeinek megfelelő termék keletkezhet. Ez olyannyira releváns, hogy bizonyos termékcsoporthoz Lengyelország forgalmának körülbelül a 40%-át teszi ki, míg ugyanez a termékcsoporthoz a magyar forgalomnak ez bőven 10%-a alatt van, így nehéz egy új

terméket bevezetni ebbe a csoportba, mert Magyarországon garantáltan kisebb rá az igény, mint Lengyelországban.

A belső együttműködésben a 7 divízió mátrix szervezatként épül fel, tehát a divíziók külön működnek, emellett különféle támogató osztályok közösek, mint a Sales, Human Resources (HR), Information Technology (IT) stb, emiatt az őket támogató vállalatirányítási rendszer is ugyanaz, ugyanazzal a logisztikai modullal. Ez újabb komplexitást idéz elő, mivel nem lehet csak az MGD igényeire specializálva alkalmazni ezeket, muszáj összehangolva használni őket, ami a divíziók külön-külön teljesítésére negatív hatással lehet.

2.2 Agilis működés a vállalaton belül

A GD egy Fast-Moving Consumer Goods (FMCG) termék kategória, ami azt jelenti, hogy relatív olcsó és relatív gyakran vásárolt termékek csoportja. Ez magában hordozza azt, hogy egy hiba a tervezésben és kivitelezésben hamar észrevehetővé válik a számokban. Ez előnnyé válhat a szigorú monitorozás alkalmazásával, mivel a hibák így elméletileg gyorsan javíthatóak. Azonban ez okozza a nehézséget is: egy hirtelen kiderült problémára a döntéshozóknak hirtelen kell reagálniuk. Ez egyrészt azért nehéz, mert, egy új tervet kellene készíteniük, amit azonban nagyon körülményes gyorsan és effektíven megtenni, idő vagy emberi erőforrás hiányában. Másrészt nem is könnyű felismerni ennek szükségességét, helyett könnyen adódik a tervek nem teljesüléséért a végrehajtás tervezés „hibáját” keresni. Tovább nehezíti ezt a feladatot a folyamatos együttműködés igénye a vállalaton és a klaszteren belül is.

Ez különösen veszélyes a GD-re nézve, mivel főként 2 termékcsoportha van: müzlik, amik a forgalom körülbelül 85%-át teszik ki, és szeletes müzlik, ami a forgalom fennmaradó 15%-át. Ugyanez a szituáció sokkal kevésbé veszélyes a konyhai készítmények divíziójára nézve, mivel ez az üzletág jelentősen több termék kategóriából áll, így, ha az egyik termékcsoportha forgalma zuhan egy hiba miatt, az a bevételnek és a portfóliónak egy jelentősen kisebb hányadát teszi ki, valamint könnyen kiegyensúlyozza a forgalmat a többi termék.

2.3 Forecast a vállalaton belül

A forecast az az adatokból álló számszerűsített információ, ami képes előre jelezni a döntéshozóknak az eladások számát a jövőben. Enélkül nem lenne képes a cég tervezni, hiszen nem tudnák előre jelezni a gyárnak, hogy milyen évi termelésre számíthatnak, mennyi csomagoló-

anyagot rendeljenek az adott hónapokra. Nem lenne lehetőségük felosztani a marketing költségvetést a divíziók között, havi szinten, termékekre lebontva. Ez képezi az alapját minden tervezésnek a vállalaton belül, és elengedhetetlen a megfelelő és kiszámítható működéshez.

A forecastolás bázis alapon történik. Ez azt jelenti, hogy amikor elkészül, akkor van egy alapszám, amiből kiindul a tervezés, ez a bázis. A tervezés folyamán résztvevő szerepkörök képviselői ezután közös munkával megbecsülik, hogy az olyan paraméterek, mint a termékre szánt promóció, marketing stratégia, sales stratégia, hogyan befolyásolják ezt a bázisszámot. Ez az alapszám, az előző éves valós eladásokból származik, tehát nem a forecastokból. Ennek előnye, hogy egy nagyságrendileg helyes becslést kapunk előző évekből kiindulva. A hátránya viszont pont ugyan ebben mutatkozik meg, mivel olyan alapértékből kell kiindulnia a tervezésnek, ami nem az aktuális piaci helyzetet veszi figyelembe, hanem a múltban lévő. Így, hiába is jön egy olyan piaci változás, ami gyökeresen megváltoztatná ezt a bázisértéket, az egész évre nem tudnak már ezzel az új számmal tervezni, de a hónapokra sem, mivel a havi forecast értékek az évi lebontása hónapokra.

Megkülönböztethető éves és havi forecastolás. Az éves forecastolás egy hónapokon át tartó tervezési folyamat, aminek célja, hogy a GD az egész évre megbecsülje előre a várható eladásokat. Ezt megteszi a klaszteren belül lévő többi ország képviselője is, és feltöltik az informatikai rendszerbe, amin keresztül a gyár is látja ezeket az igényeket, így előre tudnak tervezni, hogy van-e annyi rendelkezésre álló gép, erőforrás, hogy minden ország igényét teljesítsék. Itt is látható, hogy a közös együttműködés az európai csoport többi tagjával negatívumokkal járhat a vállalat számára, hiszen előfordulhat, hogy a kiszámított forecasthoz nem képesek tartani magukat, mert a gyár azt jelzi vissza, hogy nem tud annyi terméket legyártani, mint amennyit kért a cég. Ez újabb forecast tervezését igényli, ami újabb idő és erőforrás költség.

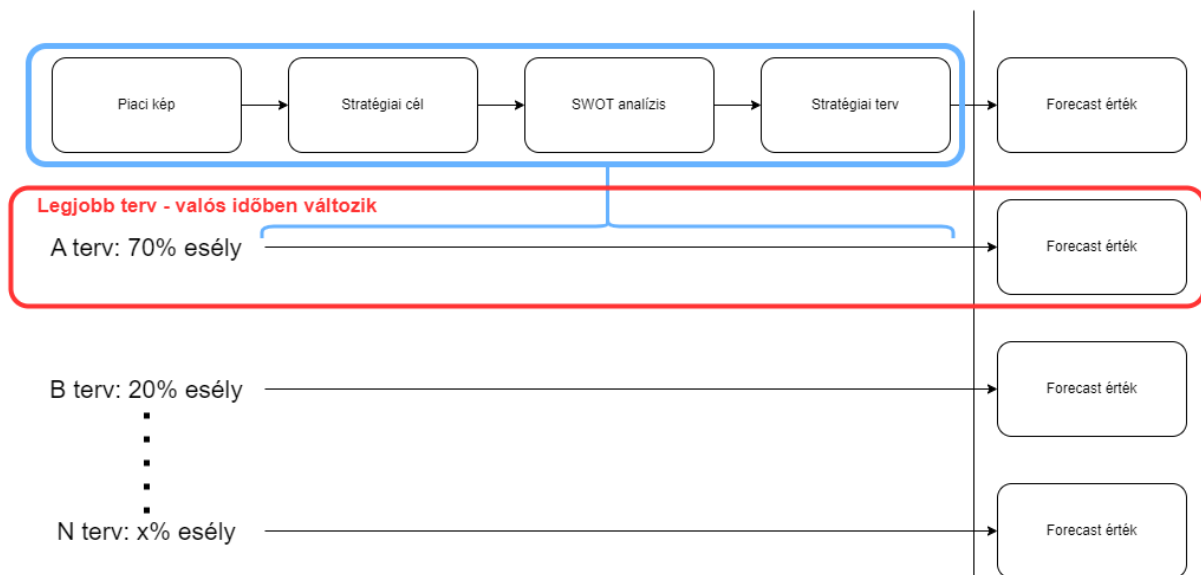
Az éves forecast tervezése május környékén kezdődik. Először ezen, a májustól szeptemberig tartó időszakon előre kell jelezniük az egész éves eladásukat forecast segítségével. Ezután szeptembertől novemberig elkészítésre kerül a havi lebontás erről a becsült évi mennyiségről. Eközben júliusban már eljuttatják a gyárnak az igényt arról, hogy milyen gépfelhasználás várható a termékek terén és mennyi csomagolóanyagra van szükség. Ezt minden egyes termékre meg kell tenni, és azok minden létező kiserelésének variációjára is.

A havi forecastolás folyamata a klaszter helyett csak az MGD-n folyik. Ez egy minden hónapban megtartott meeting sorozat, ahol felülvizsgálják az aktuális hónap és a soron következő 3

hónap forecastjait termékre lebontva és ezen belül is kiszíreelésre lebontva. Amennyiben arra a megállapításra jutnak, hogy a korábban megbecsült forecast már nem megfelelő az éppen vizsgált hónapra, azt megváltoztatják. Ezeket a havi forecastokat negyedévente revízionálják, hogy mennyire voltak sikeresek. Egy becsült érték jónak tekinthető a vállalat értékrendje szerint, ha az maximálisan 10%-kal tér el a valós eredménytől. Elfogadhatónak, de már egyáltalán nem jónak, pedig 28%-os eltérésig tekintik.

Amennyiben hirtelen változás történik a piacon, lehet ez a konkurencia egy stratégiai lépése, vagy egy globális esemény, ez új forecastot igényelhet azonnal. A jelenlegi rendszerben csupán arra van lehetőségük, hogy a jelenlegi hónapra megállapított forecastot minimálisan változtassák, azonban teljesen újat egyáltalán nem képesek készíteni.

Ezeket a nehézségeket felismerve a vállalat MGD csapata próbálkozott már egy agilisabb működéshez vezető lépéssel, ami sikeres is volt. Kidolgoztak egy stratégiát (1. Ábra, Lásd: 9. oldal) egy új termék bevezetéséhez, amiben a forecast elkészítéséhez először piaci képet vizsgálnak, ezután elkészítik a stratégiai célt, majd SWOT-analízist végeznek, végül pedig elkészül a stratégiai terv. Ezek segítségével megbecsülik a forecast értékét. Ezt a folyamatot többször megismétlik különböző fiktív gazdasági helyzetekre, így több forecasttal rendelkeznek több szituáció esetére. Ezekhez a különböző tervekhez becsülnék egy valószínűségi értéket is a bekövetkezésükre, aminek segítségével arra is tudnak készülni, hogy melyik terv bekövetkezése valószínűbb, ezáltal milyen irányba várhatóbb a forecast eltolódása. Ez a párhuzamos folyamat tervezés esélyt adott nekik arra, hogy a termék bevezetésekor tudjanak reagálni a piaci változásokra időben, késlekedés nélkül, és hogy ne legyen akadálya az új forecast készítésének és alkalmazásának az aktuális emberi és időbeli erőforrás, hiszen az új előrejelzésekkel már rendelkeztek azelőtt, hogy szükség lett volna rájuk.



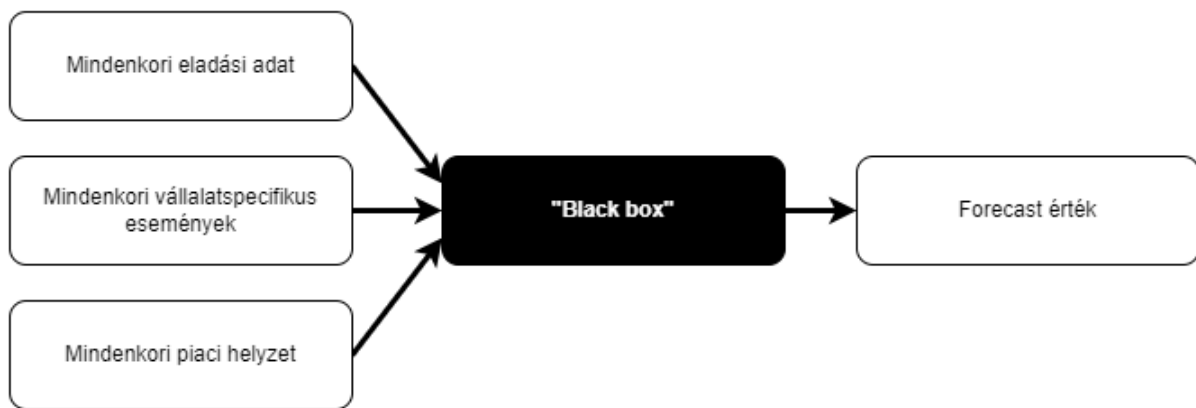
1. Ábra – A cég által tervezett stratégia Forrás: Saját szerkesztés

A stratégia eredményes volt. A termék bevezetése a GD-n belül Magyarországon volt egyedül sikeres, ami a piac gyors lekövetésének köszönhető. A kialakított stratégia egyetlen hátránya, hogy több kezdeti erőforrást és kommunikációt igényel, ami növeli a cégen belüli komplexitást, amit a nagyvállalatok tipikusan próbálnak elkerülni, mivel több hiba lehetőségnek ad helyet. A stratégia sikeressége egyúttal rávilágít a kutatás aktualitására és bebizonyítja, hogy ezen a hektikus piacon a tervezett szoftver által biztosított szolgáltatásra van igény a valós piaci felhasználásban.

3. Szakirodalmi áttekintés

3.1 Neurális hálózat alapú rendszerek

A megvalósítandó funkció és megoldandó probléma jellegéből kifolyólag a kutatás elején egy gépi tanulós megoldás került kiválasztásra [1]. Ez a típusú szoftver több indokból is megfelelő választás egy ilyen feladatra: be tud befogadni input paramétert egy felhasználótól, ami képes leírni az aktuális piaci helyzetet, ezt az inputot fel tudja dolgozni előző éves ismeretek alapján, tehát ismeri a piaci trendeket, ezáltal pedig értelmes számszerű forecast adatot tud adni a felhasználónak. (2. Ábra)



2. Ábra - A rendszer "Black box" alapú működése Forrás: Saját szerkesztés

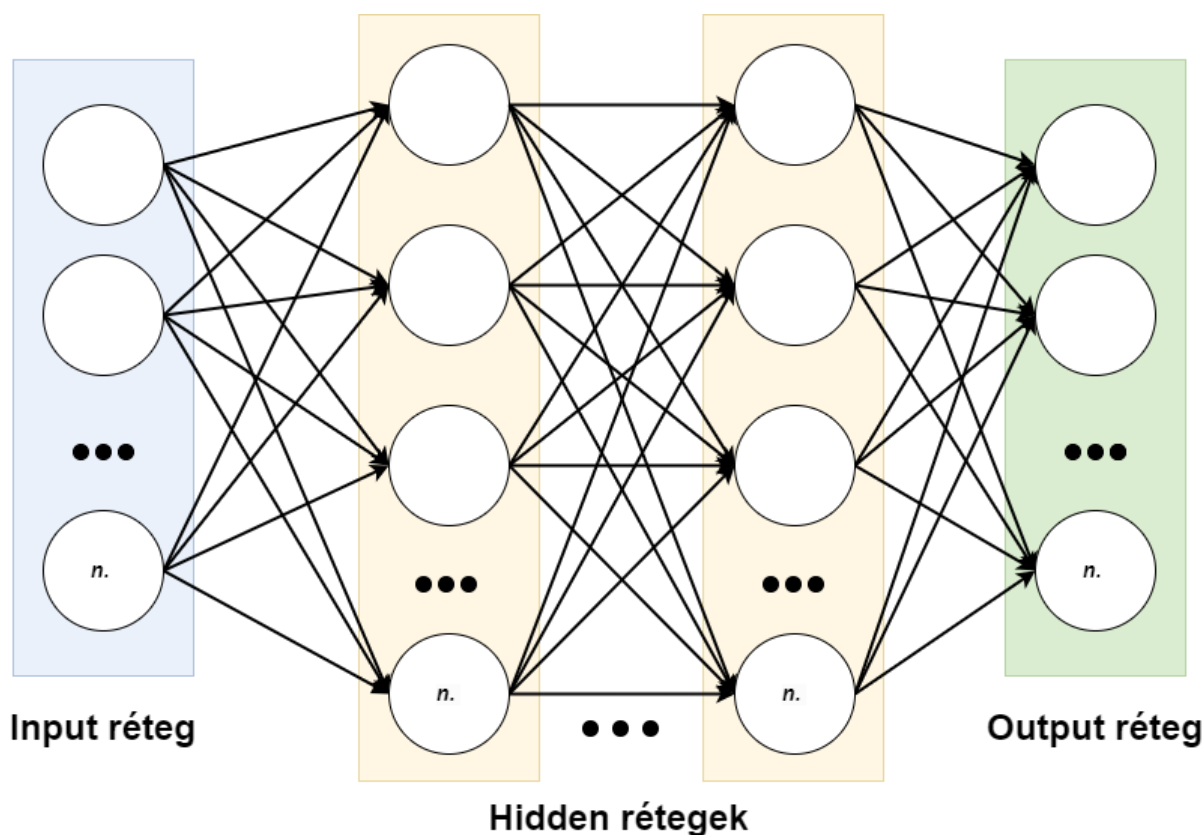
Ahhoz, hogy egy ilyen modell ismerhesse a piaci trendeket és megfelelően tudjon működni, gépi tanulásra van szükség. Ez azt jelenti, hogy a szoftvernek kialakul egy belső értékelési rendszere, ami képes úgy feldolgozni a bemeneti paramétereket, hogy a kimeneti közelítése az elvárthoz közel legyen.

Egy ilyen modell tanításához nagy mennyiségű adatra van szükség. A céltól függően ez az adat lehet kép, szöveg vagy számszerű adat [2]. A kutatás folyamán az adatokat a vállalat biztosítja, ezek a cégnek évekre visszamenően tárolva vannak az informatikai rendszerükben, az SAP-ban. Számszerű formátumban vannak tárolva, és pontosan szűrhetők egy adott évre, hónapra, hétre, illetve termékekre és ezen belül kiszerezésre is. A kutatás és a szoftver hatékonysága szempontjából ez nagyon előnyös, mivel minél több információval van ellátva a rendszer, annál pontosabb predikciókra lesz képes.

Ez a tanítás egy mély neurális hálózaton keresztül megy végbe (3. Ábra). Egy ilyen mély neurális háló az architektúrától függően kevesebb-több rétegnyi neuronból áll. A neuronok a hálózatok építőelemei, ezek felelősek az adat ideiglenes tárolásáért, illetve feldolgozásáért. A választott architektúrától függően különböző kommunikáció lehetséges ezek között az adatfeldolgozó egységek között a rétegekben belül. A rétegeket feloszthatjuk 3 fő részre:

- **Input réteg:** a bemeneti adat vektorizálásra kerül és a mélyebb rétegekben fejlődik tovább.
- **Hidden rétegek:** a megkapott vektorizált adatok feldolgozásra kerülnek az LSTM cellák által és kiszámítódik a kimenet.
- **Output réteg:** megkapja a kimeneti adatokat. Ezután inicializálhatja a hiba visszatérjesztést a súlyozások és eltolások változtatásához.

Az input réteg feladata, hogy befogadja a feldolgozandó adatokat, a kutatás esetében az SAP-ból kinyert Excel formátumban tárolt adatokat. Ezután tovább küldi az információt a hidden rétegekbe, amik számításokat hajtanak végre a kapott adatokon. A kapott eredmények az output rétegbe kerülnek a hidden rétegekből, ahol ezután készen állnak további felhasználásra, mint végeredmény a bemenő inputra.



3. Ábra - Illusztráció egy mély neurális hálózatarcitektúráról Forrás: [3] nyomán saját szerkesztés

Ha a tanítás sikeres, akkor a szoftvernek képesnek kell lennie egy bemenő adatot feldolgozni a belső számításai segítségével, és olyan kimenetet adnia, ami elvárt tőle. A kutatás esetében, tesztelhető a szoftver tanításának sikeressége, hogyha input adatként megkap egy múltból származó t időponthoz tartozó adatot, majd pedig prognosztizálja a $t + 1$ időponthoz tartozó adatot, ami összevethető a valós $t + 1$ időponthoz tartozó adattal.

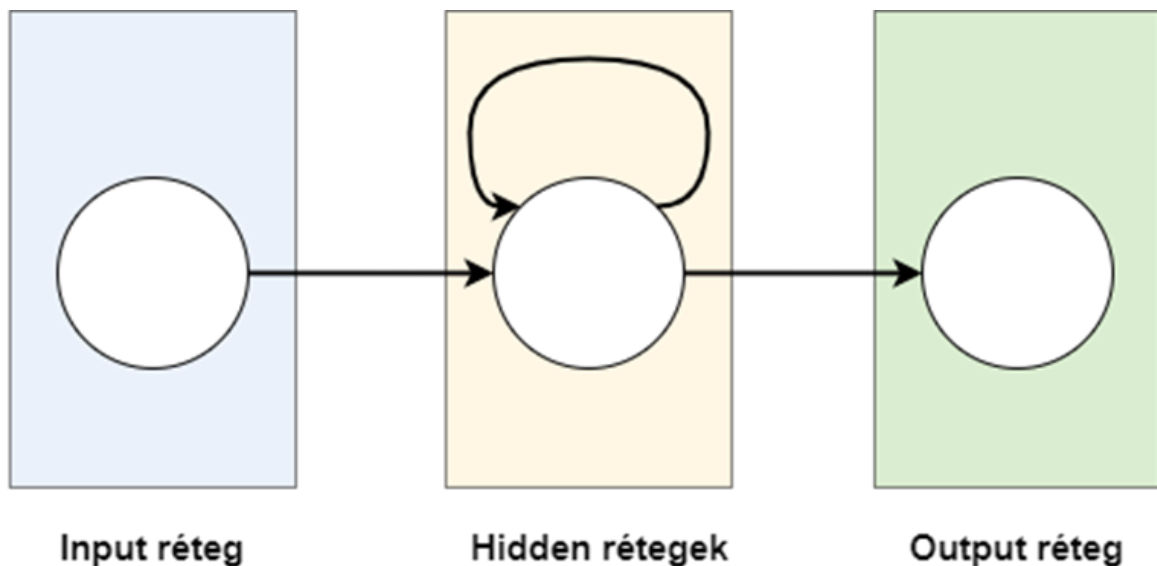
3.2 A választott architektúra ismertetése

A kutatás folyamán készített szoftvernek rendelkeznie kell azzal a készséggel, hogy figyelembe vegye az adatok időrendiségét. Ez azt jelenti, hogy meghatározó szerepe van annak, hogy egy hónap eladásainak adatai a valós sorrendben kerülnek feltanításra, vagy egy kevert sorrendben például. Ez azért jelentőségteljes, mert így tud a modell trendeket felfedezni. Ehhez a választott architektúrának képesnek kell lennie figyelembe venni az előzőleg kezelt adatokat az új adatok feldolgozása közben. Gyakorlatban ez úgy valósul meg, hogy ha egy hónap adatai feldolgozásra kerülnek az első héttől a negyedikig bezárólag, az első hét feldolgozása befolyással lesz a második, harmadik és negyedik hét feldolgozására is.

3.2.1 A választott rekurrens modell

A gépi tanulós modellek esetén ez a működés úgy érhető el, ha egy rekurrens modellt választunk (4. Ábra, Lásd: 13. oldal) [3], [4]. Elsődlegesen ez abban különbözik egy hagyományos neurális háló felépítésétől, hogy a neuronok rendelkeznek még egy önmagukba visszacsatoló éllel, amelynek úgyszint van egy saját súlyozása és eltolása. Ez azt jelenti, hogy a hidden réteg neuronjai rendelkeznek egy memóriával, ami mindig befolyásolja az éppen aktuálisan feldolgozott input adatot. Tehát például a D_{t-2} befolyásolja D_{t-1} -et, ezáltal indirekt módon befolyásolja D_t -t is, hiszen D_{t-1} befolyásolja D_t -t.

Egy adathalmazon való tanítás folyamán a visszacsatoló élek súlyozása és eltolása nem változik. Ez azt eredményezi, hogy a gradiens exponenciális növekedésbe vagy csökkenésbe kezd, attól függően, hogy a súlyozás értéke alacsonyabb vagy magasabb mint 1. Ezt az exponenciális változást felrobbanó vagy elhalványuló gradiens [5] problémának hívják. A gradiens túl nagyra vagy túl kicsire változása ellehetetleníti a backpropagation (BP) algoritmus működését, így a modell tanítását is [6], [7] a túl hosszú adatsorokon.



4. Ábra - Illusztráció egy rekurrens neurális hálózataarchitektúráról Forrás: [9] nyomán saját szerkesztés

A BP egy olyan gépi tanulási algoritmus, ami a hálózaton belüli súlyozásokat és eltolásokat a hiba visszaterjesztés segítségével közelíti az optimumhoz. Többféle módosított verziója létezik. Az időbeli adatsorok feldolgozására és a rekurrens hálózatok esetén a backpropagation through time (BPTT) nevű modifikált verzió kerül alkalmazásra a legtöbb esetben. Ez annyiban különbözik a hagyományos BP-től, hogy a tanítás időben is szétterített.

Az algoritmus működése a következő módon írható le: miután a bemeneti adatokat feldolgozza a neurális háló, a kapott kimenetet összehasonlítja a valós várt eredménnyel. Ennek a két értéknek a különbsége a hiba. Az algoritmus hátulról előre haladva, tehát az output rétegtől az input réteg irányába visszaterjeszti a hibát. A hiba mérete befolyásolja a súlyozások és eltolások változtatásának mértékét. Ha a hiba visszaterjesztése megtörtént, a neurális hálózat összes súlyozása és eltolása frissítésre került. Az értékek megváltoztatásához különböző optimalizációs technikák is alkalmazhatóak, mint például a gradient descend.

A szoftver elkészítéséhez a fentiek ismeretében, nem megfelelő egy hagyományos rekurrens neurális hálózat alkalmazása. A kutatáshoz így egy LSTM architektúra alapú modell került kiválasztásra [8]. Az LSTM architektúra lehetővé teszi az adatok szekvenciális feldolgozását, így lehetőséget adva a trendek megtanulására a modellnek, anélkül, hogy előállna a felrobbanó vagy elhalványuló gradiens problémája.

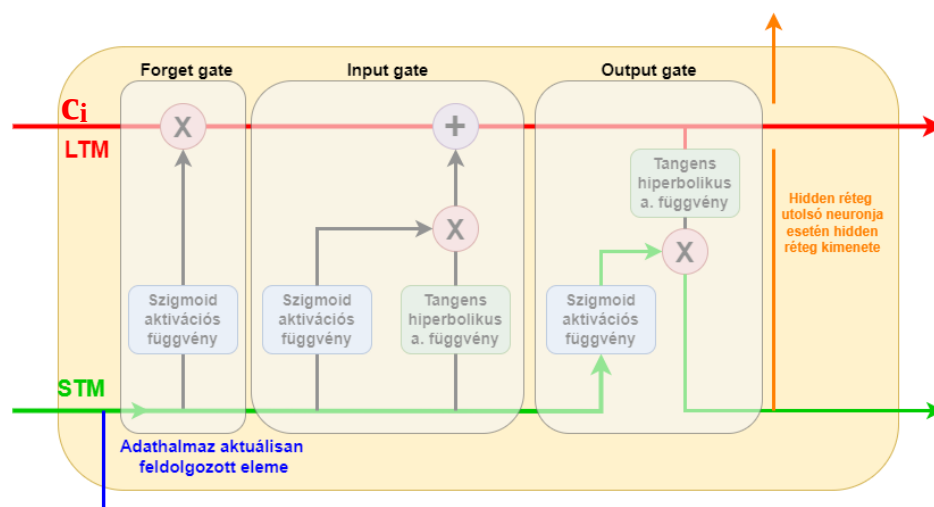
3.2.2 Az LSTM modell neuronjai

Az LSTM architektúra fő változtatása a neuronjai felépítésében található. Ezek az átalakított cellák úgy képesek feldolgozni az adatot, hogy nem kezdenek exponenciális változásba, továbbá képesek megőrizni a rekurrens hálók emlékezőképességét, tehát $Dt - n$ -ik érték továbbra is befolyásolja a Dt -ik értéket. Az ilyen viselkedés eléréséhez a neuronok egy él helyett kettőn szállítják az információt, ez a Long-Term Memory (LTM) és a Short-Term Memory (STM). Ezek az élek helyettesítik a visszacsatolást, és ezeken keresztül történik az információ áramlása a neuronok között. Az LTM célja, hogy tárolja a feldolgozott adatok fontosnak ítélt részeit, míg az STM célja, hogy meghatározza melyek a fontos részek, amiket az LTM-nek továbbra is tárolnia kell, és melyek azok az új jelentős információk, amiket hozzá kell adni ezekhez az LTM által tárolt információkhoz.

3.2.3 Az LSTM modell kapujai

Az LSTM architektúra felépítése azon kívül, hogy több élen áramlik az információ, még a neuronok belső felépítésében is különbözik (5. Ábra). Az STM kapuk felhasználásával képes eldönteni, hogy milyen információ kerüljön megtartásra, milyen információ adódjon hozzá az LTM-hez, és ennek az új információnak hány százaléka adódjon hozzá az LTM-hez. Ezek a kapuk a következők:

- Forget gate: eldönti, mely régi információkat törölje a cella állapotából
- Input gate: meghatározza, mely új információkat adja hozzá
- Output gate: kiválasztja, mely információk kerüljenek tovább az aktuális állapotból



5. Ábra - Az LSTM cella felépítése Forrás: [11] nyomán saját szerkesztés

A forget gate fogadja az adathalmaz aktuálisan bemenő input értékét, az LTM szálon érkező adatot és az STM szálon érkező adatot. Amennyiben a hidden rétegek első rétegébe tartozik ez a neuron, akkor az LTM és az STM értéke 0, mivel a neuronnak nincs megelőző hidden rétegbeli cellája. A forget gate-n belül az aktuálisan bemenő érték és az STM szálon bemenő érték külön-külön megszorozódik a súlyozásukkal, majd összeadásra kerülnek, és végén hozzáadódik az él eltolása. A kapott érték bemenetként szolgál egy szigmoid aktivációs függvénynek, amely ezt az értéket egy $]0,1[$ intervallumba eső számmá alakítja. Ez a kapott érték határozza meg azt, hogy az LTM szálon érkező információ hány százalékát őrizzük meg. Tehát, összeszorozódik a LTM szálon bejövő érték ezzel a százalékos értékkel, ezáltal frissül az LTM értéke.

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

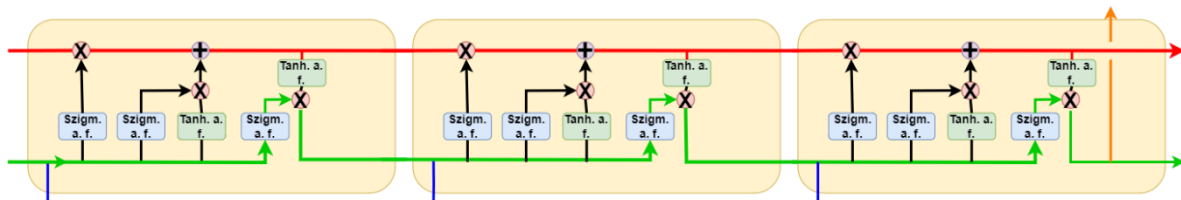
Az input gate fogadja az adathalmaz aktuális bemenő input értékét és az STM szálon érkező értéket. Ezeket felhasználva az input gate kiszámítja, hogy milyen potenciális értéket adjunk hozzá az LTM-hez, és ennek a potenciális értéknek hány százalékát adjuk hozzá. Ehhez a kapu 2 aktivációs függvénnyel rendelkezik: a már megismert szigmoid aktivációs függvénnyel és egy tangens hiperbolikus aktivációs függvénnyel. Előbbi felelős a hozzáadandó érték százalékának kiszámításáért, míg utóbbi a hozzáadandó potenciális érték kiszámításáért. Ezek bemeneti paraméterét úgy képezzük, hogy az aktuális bemeneti érték és az STM szálon bemenő érték külön-külön megszorozódik a súlyozásukkal, majd összeadásra kerülnek, és a végén hozzáadódik az él eltolása. A tangens hiperbolikus aktivációs függvény az inputot egy $] -1,1[$ intervallumba eső számmá alakítja. Ennek a két eredménynek összeszorozásával megkapjuk az LTM-hez hozzáadandó értéket, ami egyúttal az új LTM érték, amit a következő cella forget gate-je és az aktuális output gate fog megkapni, mint bemenet.

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

Az output gate fogadja az adathalmaz aktuális bemenő input értékét, az STM szálon érkező értéket és az újonnan kiszámolt LTM értékét. Ezeket felhasználva a kapu kiszámítja, hogy milyen érték haladjon tovább az STM szálon. Amennyiben ez a neuron a hidden réteg utolsó rétegében van, ez lesz az output réteg bemenete is, ha nem, akkor a következő LSTM cella forget gate-nek a bemeneti STM szálon futó értéke. Az output gate 2 aktivációs függvénnyel rendelkezik hasonlóan az input gate-hez: az első egy szigmoid aktivációs függvény, ami azért felelős, hogy kiszámítsa, hogy a potenciális érték hány százaléka adódjon hozzá az STM-hez. Ennek bemenete úgy képződik, hogy az aktuális bemeneti érték és az STM szálon bemenő érték külön-

külön megszorozódik a súlyozásukkal, majd összeadásra kerülnek, és a végén hozzáadódik az él eltolása. A második aktivációs függvény a tangens hiperbolikus, aminek bemenete az LSTM szálon futó új érték. A függvény ezt az inputot egy $]-1,1[$ intervallumba eső számmá alakítja, ami az STM potenciális hozzáadandó értéke. A két aktivációs függvény kimenetét összeszorozva, majd ezt az értéket hozzáadva az eddigi STM-hez megkapjuk az új STM szálon futó értéket. Ha az eredményt számító cella a hidden réteg utolsó neuronja, akkor ez a hidden réteg kimenete is. Amennyiben nem, akkor ez az STM szálon futó érték a következő cella forget gate-nek a bemenete (6. Ábra).

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

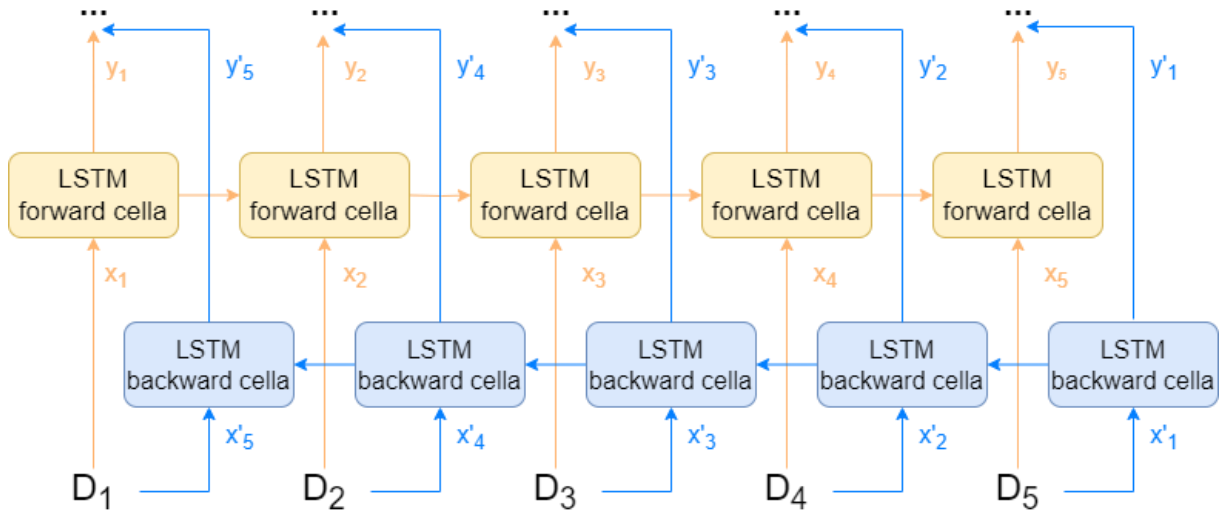


6. Ábra - Kaszkádosított cellák Forrás: [11] nyomán saját szerkesztés

3.2.4 Bidirectional LSTM

A Bidirectional (kétirányú) LSTM (BiLSTM) architektúrális felépítése jelentős mértékben megegyezik a hagyományos, egyirányú LSTM felépítésével. A különbség, hogy az információ áramlása nem csak az egyik irányba történik, jellemzően előlről hátra (például egy mondat elejétől a végéig, vagy egy hétnyi eladási adatnál hétfőtől vasárnapig), hanem hátulról előre is. Ez a változtatás lehetővé teszi azt, hogy a neurális háló ne csak múltbeli adatokat használjon fel jövőbeli előrejelzésekhez, hanem „tekintszen a jövőbe”, hogy megtudjon határozni köztes értékeket. A kutatás alatt ez a sellout adat generálás folyamán került felhasználásra, hogy a modell t időpont esetén ne kizárólag t_n pillanatbeli adatok alapján próbálja meg kiszámítani a t időponthoz tartozó értéket, hanem t_{+n} adatok felhasználásával is.[9], [10]

Ahhoz, hogy a modell rendelkezhesen jövőbeli információval egy LSTM rétegre van szükség, ami az adatsor feldolgozását hátulról kezdi meg. Így, egy 5 adatból álló adatsor esetén, ha meg akarjuk állapítani a 3. értéket, akkor a modell rendelkezésére fog állni 2 mélységű múltbeli tudás az első 2 adatból, és 2 mélységű tudás a 2 jövőbeli adatból. (7. Ábra, Lásd: 17. oldal)



7. Ábra - A BiLSTM működése. Forrás: Saját szerkesztés

Az architektúrán kívül a neuronok közötti adatáramlás, tehát a súlyozások és eltolások számítása, az LSTM neuronok belső működése, továbbá a BPTT működése megegyezik a fentebb, hagyományos LSTM-nél leírtakkal.

Ami különbséget jelent még, az adott t időponthoz tartozó cellák értékének meghatározása, hiszen nincs egyértelmű kimenet t időponthoz, mivel ahhoz 2 cella értéke tartozik. A fenti ábrát példaként véve, a D_3 adathoz és t_3 időponthoz kimeneti érték az y_3 és az y'_3 is. Ahhoz, hogy egyértelmű kimenete legyen egy időpillanatnak, a két kimeneti vektort (tehát y_{3-t} és y'_{3-t}) konkatenálni kell. Ez azt jelenti, hogy a 2 vektort össze kell vonni, ami egy harmadik, konkatenált vektort eredményez, ami információ tartalmaz a múltból és a jelenből is. Legyen egy BiLSTM forward rétegének első eleme F_1 , míg B_n a backward réteg utolsó eleme. Ez azt jelenti, hogy a múltból 1 adat információjával rendelkezik az adott időpillanatban a neuron pár, míg a jövőből n adat információjával, ahol n az adatsor összes elemének mennyiségét jelöli. Legyen K_1 a konkatenált vektor, így ez:

$$F_1 = [f_1, f_2]$$

$$B_n = [b_{n-1}, b_n]$$

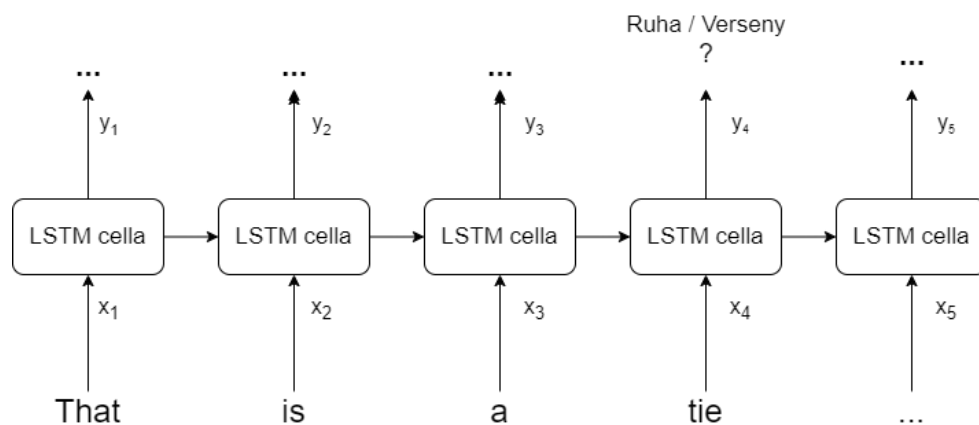
$$K_1 = [f_1, f_2, b_{n-1}, b_n]$$

A következő példa egy természetes nyelvi feldolgozáson keresztül mutatja be a működését egy BiLSTM-nek.

That is a tie, a clothing worn at formal occasions.
 (Ez egy nyakkendő, egy ruhadarab, amit hivatalos eseményeken viselnek)

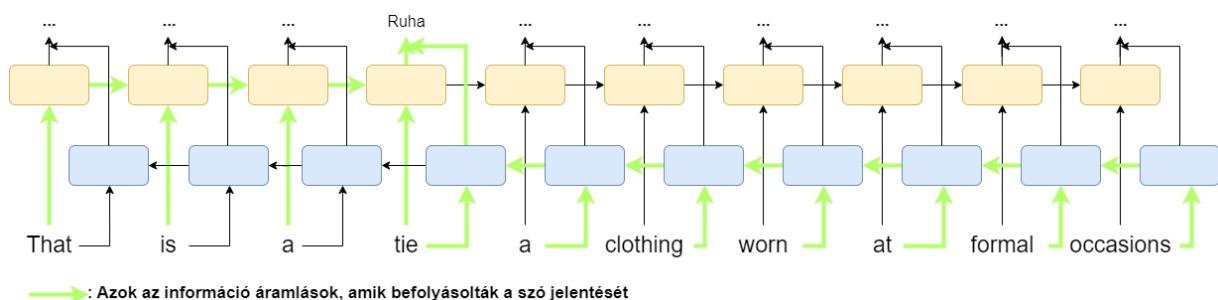
That is a tie, the tournament can't be decided yet.
 (Ez döntetlen, így nem lehet befejezni a tornát)

Ha a modell egy feladata az, hogy meghatározza a következő szó jelentését a mondat folyamán kizárólag múltbeli információra támaszkodva, akkor a sikerességének esélye csekély lesz a „That is a,” mondatrész után, mivel a „tie” egy többértelmű szó. Ahhoz, hogy meghatározza a jelentését csakis az előtte álló „That is a” mondatrész áll rendelkezésére, ami nem ad specifikus tudást a modellnek ahhoz, hogy eldöntse, hogy egy ruhadarabról vagy versenyformáról van szó. (8. Ábra)



8. Ábra - A mondat feldolgozása hagyományos LSTM segítségével. Forrás: Saját szerkesztés

Ahhoz, hogy a modell információval rendelkezessen afelől, hogy a „tie” szó melyik jelentése aktív a mondatban, több információra van szüksége. Ehhez a fentebb leírt és bemutatott módon egy backward réteggel kiegészítve a rendszert, további tudáshoz jut a modell, ami által képes lesz meghatározni (feltételezve, hogy megfelelően volt tanítva), hogy egy adott szövegkörnyezetben a szó melyik jelentése aktív. (9. Ábra)



→ : Azok az információ áramlások, amik befolyásolták a szó jelentését

9. Ábra - A BiLSTM működése nyelvi feldolgozásnál. Forrás: Saját szerkesztés

3.3 Természetes nyelvi feldolgozás

Az NLP egy általános összefoglaló kifejezése a természetes nyelv feldolgozásnak, ami az utóbbi időben elsődlegesen gépi tanulás alapú megközelítéseket jelentett. Ez a kifejezés foglalja össze egyúttal a humán és gépek közötti értelmes kommunikációt, amelyben a gép megpróbálja értelmezni az emberek által közölt szöveget, kontextusba helyezni azt, majd pedig ennek megfelelően kommunikációt létesíteni.

3.3.1 Szavak értelmezése gépi tanuláson keresztül

Ahhoz, hogy a fentebb leírt kommunikáció lehetségessé váljon, egy közös nyelvre van szükség a számítógépek és emberek között. Mivel a számítógépek számokon keresztül képesek végrehajtani műveleteket, az emberi szöveget először számszerűsíteni kell.

A legegyszerűbb megközelítés, a szavak hagyományos adatbázisban való tárolása, ahol minden szóhoz egy kulcsérték rendelődik. Ennek a fő hibája, hogy nem alakul ki logikai kapcsolat olyan szavak között, amik között egyébként van. Egy szótár esetén például, ábécé sorrendben ellátva minden szót egy egyedi azonosítóval (tehát sorszámozva 1-től n -ig) a „kutya” és „macska” szó teljesen eltérő azonosítót kapnának, annak ellenére, hogy logikailag összefüggőek, azonban a szótárban messze állnak egymástól.

Egy gépi tanulásra jellemző másik megközelítés a One-Hot Encoding (OHE) alkalmazása. Az OHE esetén minden szó 0-ák és egy darab 1-es variációjaként szerepel, így az első szó OHE után $n = 4$ esetén, ahol n a szavak mennyisége „1000” lesz (1. Táblázat). Ennek a hibája továbbra is az, hogy nem tartalmazza a logikai kapcsolatot 2 logikailag egybefüggő szó között. További hátránya, hogy n elemű szó esetén n mennyiségű vektort kell felhasználni a szavak tárolására, ahol minden vektor n méretű, ami jelentős tárhely felhasználást, és lassított számítási folyamatokat jelent.

	1. szó	2. szó	3. szó	4. szó	...	n . szó
szó	1	0	0	0	0	0
szó	0	1	0	0	0	0
szó	0	0	1	0	0	0
szó	0	0	0	1	0	0
...	0	0	0	0	1	0
n . szó	0	0	0	0	0	1

1. Táblázat - Az OHE bemutatása Forrás: Saját szerkesztés

A jelenleg legelterjedtebb és legeredményesebb megközelítés a vektor embeddingek (vektor „beágyazások”) alkalmazása. A működésük leírására egy sok (akár több száz) dimenziós vektor tér a legalkalmasabb, ahol a különböző tulajdonságok leírása történik [0, 1] között. Egy példa keretein belül, legyen egy vektor tér 8 dimenziós, és legyen a vektorizálandó szavak listája a következő: alma, banán, nap. Legyenek az attribútumok (tehát a dimenziók, amiken az értékelés történik) a következők: élelmiszer, növény, égitest, fényes, sárga, piros, kerek, hosszúság. Az attribútumok értékelése történjen úgy, hogy az 1-hez közel lévő értékek jelentik azt, hogy az attribútum jellemző a szóra, míg a 0-hoz közelebbi érték, hogy nem. Látható, hogy az alma és banán vektora ugyan nem megegyező, de egymáshoz matematikai szempontból közel álló vektorok, így egymáshoz logikailag közel álló jelentéssel bírnak. Az is megfigyelhető, hogy ugyan a nap mindkettő másik szóval rendelkezik hasonló értékekkel, mégsem mondható az, hogy logikailag kapcsolódna hozzájuk, vagy hogy matematikai értelemben közel állóak a vektorjaik. Az Euklideszi távolság képlet segítségével kiszámolva az „alma” és „banán” szó távolsága a vektor térben 1.61 egység, míg az „alma” és „nap” szó távolsága 2.04 egység. Növelve az attribútumok mennyiségét, ez az érték még jobban reprezentálná a logikai kapcsolatot a szavak között. (2. Táblázat) A valóságban egy vektor embedding esetén ez a dimenzió nem 8 skálán mozog, hanem akár 500-on, ami lehetővé teszi, hogy minden szó megfelelően legyen logikailag reprezentálva. [11], [12]

	Alma	Banán	Nap
Élelmiszer	1	1	0
Növény	1	1	0.2
Égitest	0.05	0.05	1
Fényes	0.2	0.1	0.95
Sárga	0.25	1	0.9
Piros	1	0	0.2
Kerek	1	0	0.9
Hosszúság	0	0.8	0

2. Táblázat - A vektor embedding bemutatása Forrás: Saját szerkesztés

A szavak közötti műveletek is alátámasztják a vektor embeddingek megfelelőségét a jelentések leírására. Amennyiben vesszük a „király” szó vektor értékét, hozzáadjuk a „nő” szó vektor értékét, továbbá elvesszük a „férfi” szó vektor értékét, akkor a kapott vektor a legközelebb a „királynő” szó vektorjához fog állni.

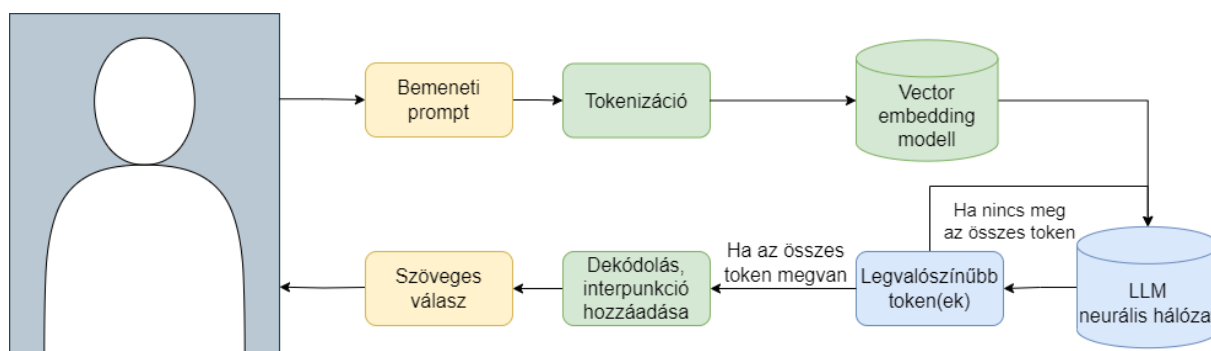
A fentebb leírt működések természetesen akkor valósulnak meg, ha megfelelő módon kerültek tanításra a vektor embedding modellek. Ezeknek a tanítása hatalmas mennyiségű folyószövegen történik, ahol a szavak közötti logikai kialakulás elsődlegesen a szavak kontextusától függ.

Minél többször fordul elő egy szó egy másik közelében, annál inkább logikailag összekapcsol-
tak lesznek.

3.3.2 Nagy nyelvi modellek

Az LLM-ek olyan algoritmusok, amelyek mögött egy feltanított neurális háló áll. Ezeknek a neurális hálózatoknak változó paraméter számaik vannak, pár milliárdostól egészen több száz milliárdokig. A tanításukhoz hatalmas mennyiségű tárhely szükséges az adathalmaz tárolásá-
hoz, továbbá úgyszint nagyon nagy mennyiségű számítási kapacitás szükséges. Ehhez jellem-
zően több tíz-százezer magas teljesítményű videokártya összehangolt működésére van szükség,
akár heteken keresztül [13]. Ezek mellett a feltételek mellett, a kutatás nem terjed ki nyelvi
modellek tanítására a dolgozat ezen érettségi fokán.

Egy LLM kommunikációját egy felhasználóval a következő folyamat írja le: a felhasználó küld
egy promptot (tehát egy szöveget) a nyelvi modell felé. Ezt a promptot először tokenizálni kell.
A tokenizálás azt jelenti, hogy a szöveget szavakra, vagy szó részekre bontja egy rendszer.
Ezután ezeket a tokeneket vektorizálni kell egy előre feltanított vektor embedding modell se-
gítségével, a fentebb már tárgyalt módon. Ezután ezek a vektorok bemenetet képeznek az LLM
mögötti neurális hálónak, aminek a kimenete egy valószínűségi eloszlás, ahol minden kimenet
egy tokenet reprezentál. Tehát meghatározza, hogy melyik token a legvalószínűbb, hogy logiku-
san következne az előzőek után. Miután összeáll a szöveg a tokenekből, ezeket dekódolni kell,
tehát vissza kell alakítani a vektorokat szöveges tokenekké. Ezután következik az interpunkció
(írásjelek és mondat tagoló eszközök) hozzáadása, illetve a szavak sorrendjének meghatáro-
zása. Miután a szöveg készen áll, ez visszajuttatásra kerül a felhasználóhoz (10. Ábra).



10. Ábra - Az LLM-ek működésének ábrázolása Forrás: Saját szerkesztés

Az LLM-ek felhasználása jelenleg két formában érhető el: olyan online szolgáltatások igénybe vételével, mint például az OpenAI cég által szolgáltatott ChatGPT, ahol a felhasználó egy web interfacen (internetes felületen) vagy egy API híváson keresztül éri el az OpenAI szerverein futó modellt, ami aztán a generált választ ugyanezen a felületen szolgáltatja vissza a felhasználónak, vagy pedig API hívás esetén a választ fogadó függvényen keresztül juttatja vissza a felhasználóhoz.

A másik megközelítés a lokális futtatása az olyan a nagy nyelvi modelleknek, amelyek nyílt forráskódúak. Ezek rendszerint nagyon magas erőforrás igényvel rendelkeznek, és az átlagos otthoni környezet nem elég egy ilyen modell megfelelő gyorsaságú futtatásához. Egy kisebb modell, futtatásához javasolt a 12GB VRAM, azonban egy nagyobb, jelenleg state-of-the-art modell futtatása, mint például a Llama 3.1 24GB VRAM-ot is igényelhet.[14], [15]

3.3.3 Lemmatizáció és stemmelés

Mind a lemmatizáció (alapalak megkeresése) és a stemmelés (szótövesítés) a természetes nyelvi feldolgozás alapvető eszközei. A lemmatizáció olyan nyelvi előfeldolgozási eljárás, amely során egy szó különböző alakjait a nyelvtanilag helyes alapalakjukra (lemmájukra) redukálják, figyelembe véve a szó nyelvtani szerepét és jelentését, gyakran lexikonokat és morfológiai elemzést használva. Ezzel szemben a stemmelés egyszerűbb algoritmusokat alkalmaz, amelyek mechanikusan levágják a szó végződéseit, így egy „szótövet” hozva létre, mely nem feltétlenül a nyelvtanilag helyes alapalak.[16]

Így a lemmatizálás folyamatán keresztül, az „evés”, „eszem”, „eszel” szavak mind az „enni” ige alapalakra változik.

Így a stemmelés folyamatán keresztül, a „gyorsabbá” szónak a helyes alapalakja a „gyorsabb” lenne, azonban ez a „gyors” tövé redukálódik.

A kutatás folyamán lemmatizálás került alkalmazásra, mivel a használat során előnyösebbnek bizonyult a szavak alapalakjának kezelése, mint a tövesített formáké.

3.3.4 Szentiment analízis

Az emberi nyelv elemzésére használt kifejezés a szentiment analízis, ami tulajdonképp érzellem vizsgálatot jelent, aminek a célja, hogy meghatározza egy emberi szöveg hangvételét[17]. Felhasználása széleskörben elterjedt az informatikában, például szociális média elemzésekhez politikai posztokhoz, ahol több ezer kommentet vizsgálhatnak meg egy politikus kijelentéséhez kapcsolódóan, ezzel olyan információt gyűjtve például, hogy egy választói közegben milyen kijelentésekről vannak pozitív véleménnyel a választók, illetve olvasók és milyenekkel nem.

A szentiment analízisnek több megközelítési fajtája is létezik, ezek jelenleg: szabály rendszer alapú szentiment analízis; gépi tanulás alapú szentiment analízis; természetes nyelvi feldolgozás alapú szentiment analízis. Ezek vizsgálatához vegyük a következő szöveget példának: „Ez a gabonapehely rémesen finom! Sokkal jobb, mint amire számítottam”.

A szabály rendszer a legrégebbi megközelítés és a legegyszerűbb is. A szövegek elemzéséhez egy pozitív és egy negatív lexikont alkalmaz. Ezek a szótárak manuálisan kerülnek összeállításra, és használatonként módosíthatóak, hiszen különböző felhasználásokban 1-1 szó eltérően lehet pozitív, de máshol negatív jelentésű is. A mondat feldolgozása folyamán olyan szavakat keres, amik megtalálhatóak valamelyik lexikonban. Azok a szavak amik a pozitív lexikonban vannak meghatározva, egy pozitív pontszám van hozzájuk rendelve. Mikor feltűnnek a szövegben, a feldolgozott szövegrész pozitív pontot kap. Ennek megfelelően a negatív szavak a negatív lexikonban vannak eltárolva, így negatív pontot kap a szövegrész, mikor feltűnik egy negatív szó. Ennek megfelelően a példában a „rémesen” egy negatív jelentésű szó, míg a „finom” és „jobb” egy pozitív értékű szó. Így, ha feltételezzük, hogy mind a három szó egyenlő mértékben pozitív és negatív (tehát az értékük +1 és -1), a skála pedig -3-tól +3-ig terjed, ahol -3 a legnegatívabb érték és +3 a legpozitívabb, akkor $1+1-1 = 1$ pontos pozitivitású a mondat.

Látható, hogy ez téves végső megállapítás, mivel a mondat teljes mértékben pozitív, 3 pontos értékelést kellett volna kapnia. Ezen felül, mivel valószínűleg egyik lexikon sem tartalmazza a „mint amire számítottam” mondatrész elemeit, így ennek az értékelése teljesen kimarad, pedig ez a mondatrész azt indikálja, hogy annyira jó volt a gabonapehely, hogy az eredeti várakozásokat is felülmúlta, ami tovább kellene fokozza a szövegrész pozitivitását.

A következő módszer a gépi tanulás alapú szentiment analízis. Egy klasszikus gépi tanulási algoritmus, például egy Naive Bayes-osztályozó vagy SVM modell [18]előre meghatározott

tanulási folyamat során ismeri meg a szavak súlyozását és az összefüggéseket. Egyik előnye, hogy a mondat minden szavát értékelheti, és összefüggéseket találhat köztük, így nem csak azok a kifejezések kerülnek kiértékelésre, amik megtalálhatóak a lexikonokban.

A tanulási fázisban a modell megtanulhatta, hogy a „finomabb” és „sokkal” kombinációja valószínűleg pozitív szentimentet tükröz, de az már kérdéses, hogy a „rémesen finom” kifejezést tudja-e úgy értelmezni, mintha az lenne, hogy „nagyon finom”. Itt a modellnek már van lehetősége megtanulni azt is, hogy a mondat magában hordozza az „elvárás fölötti jellegét”, hiszen ismerheti a „számítottam” és a „finom” szó gyakorisági eloszlását.

A harmadik és legújabb módszer az LLM irányú megközelítés. Egy nagy nyelvi modell jobban megérti a mondat teljes kontextusát és a szókapcsolatok árnyaltságát, így értelmezi azt is, hogy a „sokkal finomabb, mint amire számítottam” egy fokozottan pozitív érzést fejez ki, mivel az elvárások feletti élményt jelöl. Ezenkívül, a state of the art (a kornak megfelelően a legfejlettebb) modellek könnyedén értelmezhetnek szarkazmust a mondatokon belül, így egy olyan kijelentés, mint például, hogy „Hát ez a müzli aztán jó nagyon jó volt mondhatom...” egy cikk szövegtörzsében, további kontextussal ellátva szinte egyértelműen negatív, szarkasztikus kijelentés.

Egy további előnye az LLM irányú megközelítésnek, hogy azon túl, hogy értékeli a szöveget pozitív/negatív értelemben, még összegezheti is ezt a felhasználó számára, továbbá további szűrést tesz lehetővé, amivel képes eldönteni, hogy tényleg releváns-e az adott cikk a nagyvállalat számára, vagy tévesen került kiválasztásra korábban.

3.3.5 RAG rendszer

A RAG rendszer tulajdonképp egy kiegészítő keretrendszer, ami az LLM-ek hibáit/gyengeségeit próbálja javítani. Az LLM-ek elsődleges hátránya a már fentebb említett rettentően magas számítási kapacitás, ami szükséges a tanításukhoz. A nyelvi modellek csakis olyan tudással rendelkeznek, amilyen tudás a tanító adathalmazban megtalálható volt. Ez azt jelenti, hogy egy LLM egy tanítás után, tulajdonképp egy az adathalmaz egy időpillanatbeli tudásával rendelkezik, és semmi többel, egészen ameddig újabb tanításra nem kerül sor, ami nagyon költséges. További hátrány még, hogy a modellek paraméter száma direkt módon befolyásolja a modellbe összegyűjthető tudást. Minél több paraméterrel rendelkezik egy modell, annál nagyobb tudást

gyűjthet, de annál költségesebb a tanítási folyamat. Erre megoldást jelenthet egy kisebb paraméter számmal rendelkező modell tanítása. Ennek azonban a hátulütője az, hogy minél többet tanul, annál inkább elfelejti a kezdeti tanulás állapotát (tehát azt a tudást, amivel akkor rendelkezett, mikor leállították az utolsó tanulási fázist). Ez amiatt van, mert a kis paraméter szám mellett limitált súlyozásokkal és eltolásokkal rendelkezik, így egy kisebb modellt nem lehet a végtelenségig tanítani.

A RAG keretrendszer ezekre a hibákra kínál megoldást, továbbá előnye az is, hogy lehetővé teszi, hogy az LLM adott terület(ek)en specializálódjon. Az LLM-ek nem kapcsolódnak adatbázisokhoz, kizárólag azzal a tudással dolgoznak, ami el van raktározva a súlyozásaikkal. A RAG lehetővé teszi, hogy az LLM kommunikációt létesítsen egy vektor adatbázissal, amiben technikailag korlátlan információt tárolhatunk vektorizált formában. Ez több problémát is megold, egyrészt nem szükséges az LLM-et egyfolytában tanítani, amikor új fontos tudás válik elérhetővé a számára, elég ezt hozzáadni a vektor adatbázishoz, amihez az LLM hozzáfér, ezzel kiküszöbölve a tanítás folyamatos erőforrás igényét. Másrészt, csökken a paraméter számból származó limitáció, ami eddig nagy mennyiségű paraméterek esetén az erőforrásigény miatt volt akadály, alacsony mennyiség esetén pedig az eddigi tudás potenciális elfelejtése miatt. [19]

További előnye még a RAG rendszereknek, hogy kiegészíthetik az LLM-ek generális tudását vállalat specifikus tudással. Tehát, ha a cég rendelkezik valamilyen különleges sajátossággal, ami bizonyos módon befolyásolja a hírek értékelését, akkor az LLM már nem csak a saját tudásával (ami a súlyokból származik), nem csak a gazdasági szakirodalom tudásával (ami a vektor adatbázisból származik) képes értékelni a híreket, hanem a cég saját értékrendje szerint is, ami különbözhet egy másik cég értékrendjétől, hiszen az LLM és a kiegészített gazdasági tudás nem feltétlen elég ahhoz, hogy helyesen értékelje a híreket, mivel ugyanaz a hír különbözően hathat *A* és *B* cég eladásaira.

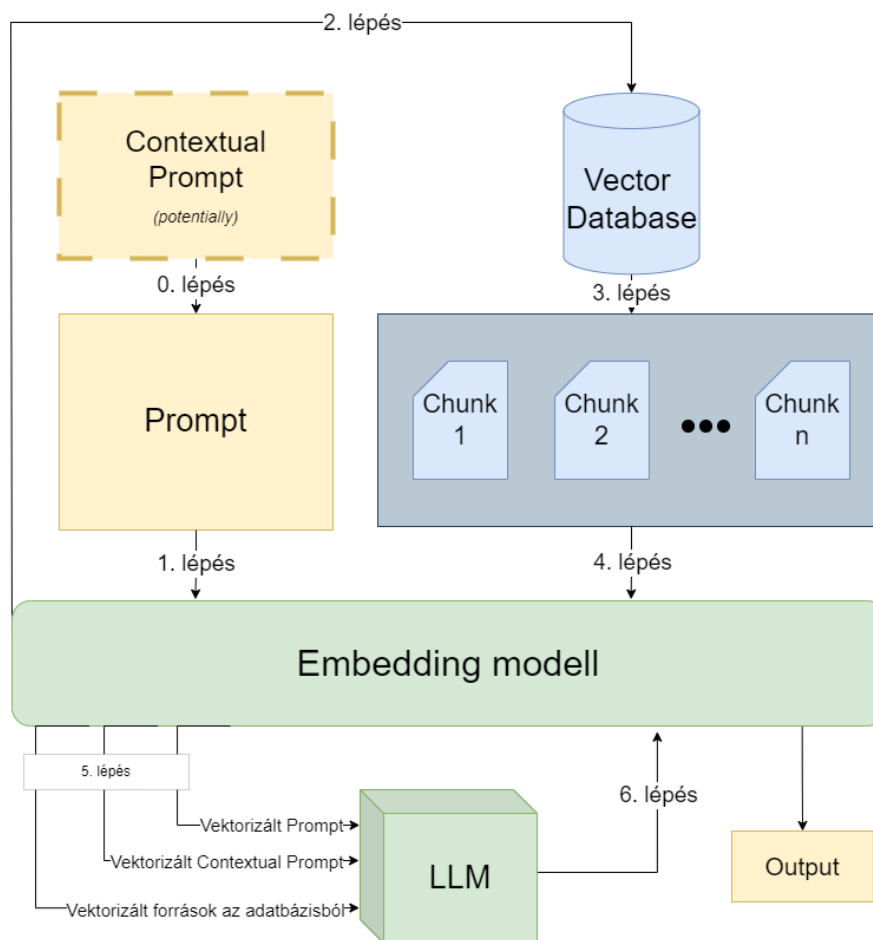
A RAG rendszer architektúrális működése hasonló egy hagyományos LLM rendszeréhez. Miután megérkezik a feldolgozandó prompt, az átmegy egy vektor embedding modellen. A tokenizáció és vektorizáció után a kapott vektor alapján keresés hajtodik végre a vektor adatbázisban. Az adatbázisban az információk „chunkok”-ban vannak tárolva, tehát lényegében szöveg tömbökben. A keresésnek több fajtája létezik, de az egyik legjellemzőbb a „semantic search”, ahol a már tárgyalt vektor embeddinghez hasonlóan, nem kulcsszavak alapján történik a keresés,

hanem a mondat/szöveg jelentése alapján. Ezután különböző módszerekkel sorrendiség alakul ki a talált chunkok között, akár a vektorok távolsága, vagy a bezárt koszinusz szöge alapján.

$$\cos \theta = \frac{\vec{a} * \vec{b}}{|\vec{a}| * |\vec{b}|}$$

Ezután az LLM nem csak az eredeti promptból készült tokenizált vektorokat kapja meg, amire válaszolnia kell, hanem kontextust is kap hozzá a vektor adatbázisból a kiválasztott legrelevánsabb vektorizált chunkok formájában. Ezekből a chunkokból az(oka)t és annyit használ fel, amennyit a szoftver architektúrája megállapít. Így, az LLM további tanítás nélkül hozzájutott további információkhoz. (11. Ábra)

Jellemzően a prompt kiegészítésre kerülhet úgynevezett „contextual prompt” -al, aminek a célja az LLM viselkedésének további irányítása a fejlesztők által, amivel plusz információval látják el az LLM-et a felhasználóról, a feladatáról, az elvárt formátumú válaszról stb.

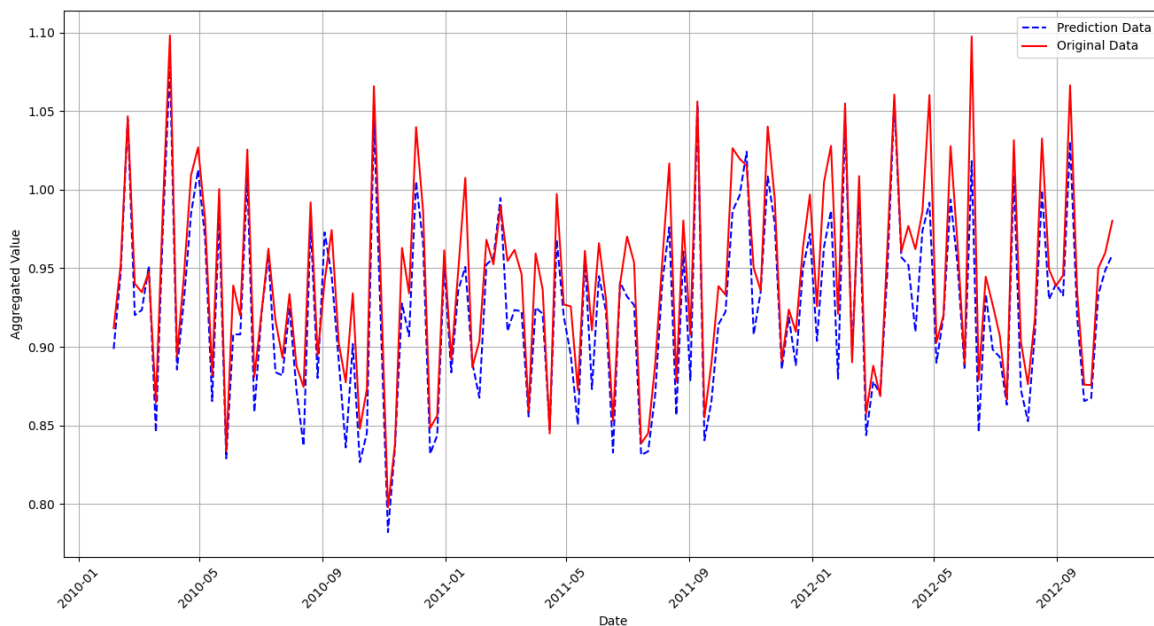


11. Ábra - A RAG rendszer általános működésének bemutatása Forrás: Saját szerkesztés

4. Prototípus szoftver

4.1 Directional LSTM előrejelzés nem vállalati adatokon

2023 őszi félévében a tudományos diákköri konferencia (TDK) alkalmával elkészült dolgozat a 2024 tavaszi félév prototípusának szolgált. Az első dolgozat célja, hasonló volt, mint a másodiknak, azonban még nem a valós vállalati adatokkal dolgozva, hanem egy előre megtisztított, rendszerezett és a szerző által már felülvizsgált hasonló adatbázissal. Az első dolgozat bemutatta azt, hogy az LSTM architektúra alapú neurális hálók alkalmasak arra, hogy előrejelzéseket nyújtson eladási adatokról (12. Ábra). Az ottani eredmények alapján került kiválasztásra az LSTM architektúra a második dolgozathoz.



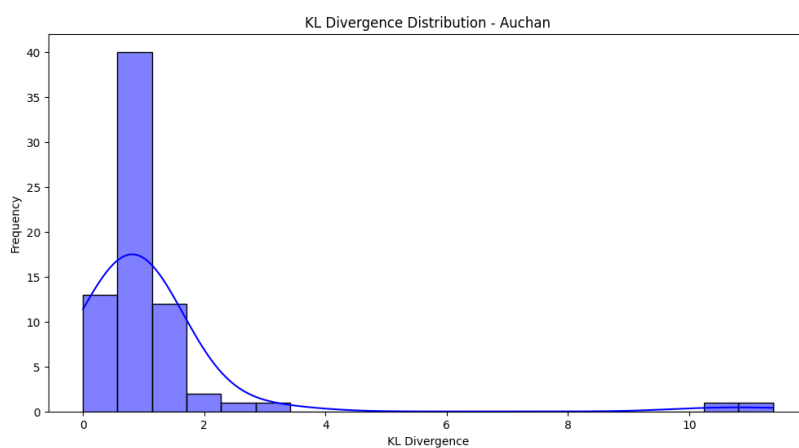
12. Ábra - A prototípus szoftver valós és előrejelzett adatai egymáshoz viszonyítva Forrás: Saját szerkesztés

4.2 Bidirectional LSTM előrejelzés vállalati adatokon

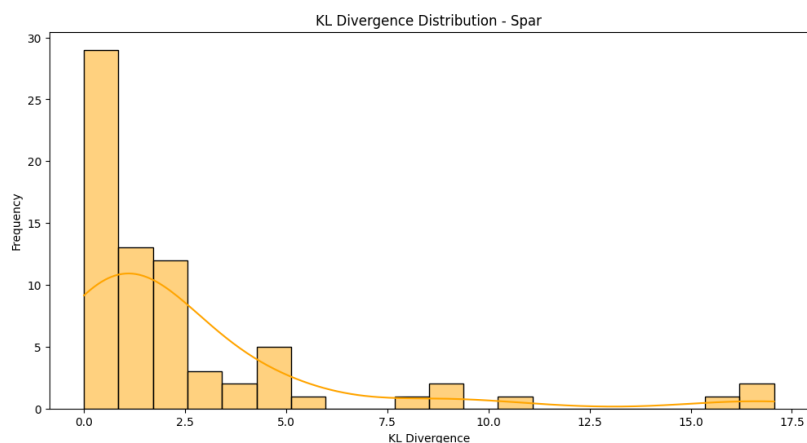
A második dolgozat jól körüljárja az adatfeldolgozás problémáját is, mivel ez nem jelentett kihívást az első dolgozat folyamán, azonban a valós nyers adatok értelmezése, tisztítása és feldolgozása során világossá vált, hogy ez a tényező nagyban befolyásolja a feltanítás és előrejelzés minőségét is.

A második TDK dolgozat alkalmával elkészült az első fejlesztett prototípus szoftver, ami már ténylegesen a nagyvállalati gazdasági adatokat alkalmazta. A vállalat 22 viszonteladó üzletnek értékesít, és ebből 3 olyan üzlet van (Tesco, Spar, Auchan), amelyek szolgáltatnak visszafele is sell out adatokat a vállalat számára. A belső és külső eladások kilóban értendők. Az előrejelzés alapötlete is ehhez kapcsolódik, mivel megvizsgálásra került, hogy ezeknek a szolgáltatott sell out adatoknak milyen eloszlásuk van, és abból indult ki a kutatás, hogy ha az eloszlás megegyezik, akkor feltételezhetően a többi viszonteladó eladásának az eloszlása is hasonló tendenciát fog mutatni.

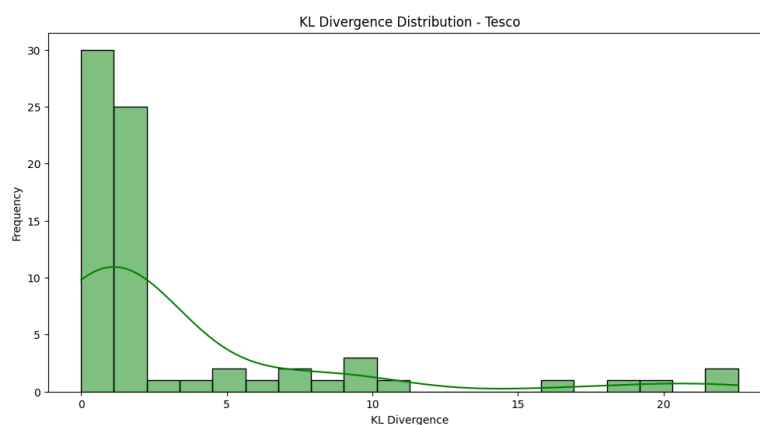
Elsődlegesen a sell in és sell out adatok aránya került megvizsgálásra minden termékre a fentebb említett Auchan (13. Ábra13. Ábra), Spar (14. Ábra, Lásd: 29. oldal) (15. Ábra, Lásd: 29. oldal), Tesco (16. Ábra, Lásd: 29. oldal) boltokban Kullback–Leibler divergencia (KL Divergencia) [20] segítségével, hogy megbizonyosodjunk arról, hogy a sell out adatok generálása lehetséges-e.



13. Ábra - Auchan eladások Kullback-Leibler [20] eloszlása Forrás: Saját szerkesztés

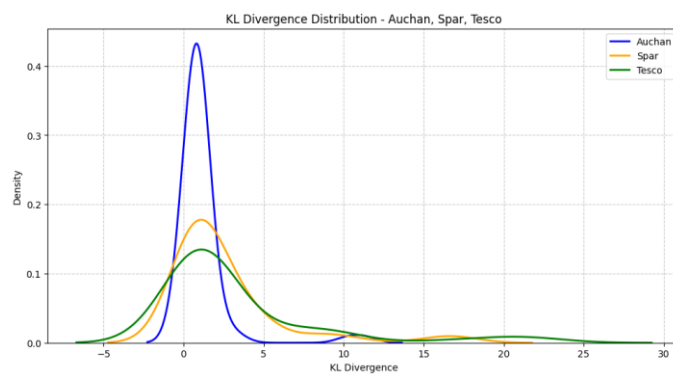


14. Ábra – Spar eladások Kullback-Leibler [20] eloszlása Forrás: Saját szerkesztés



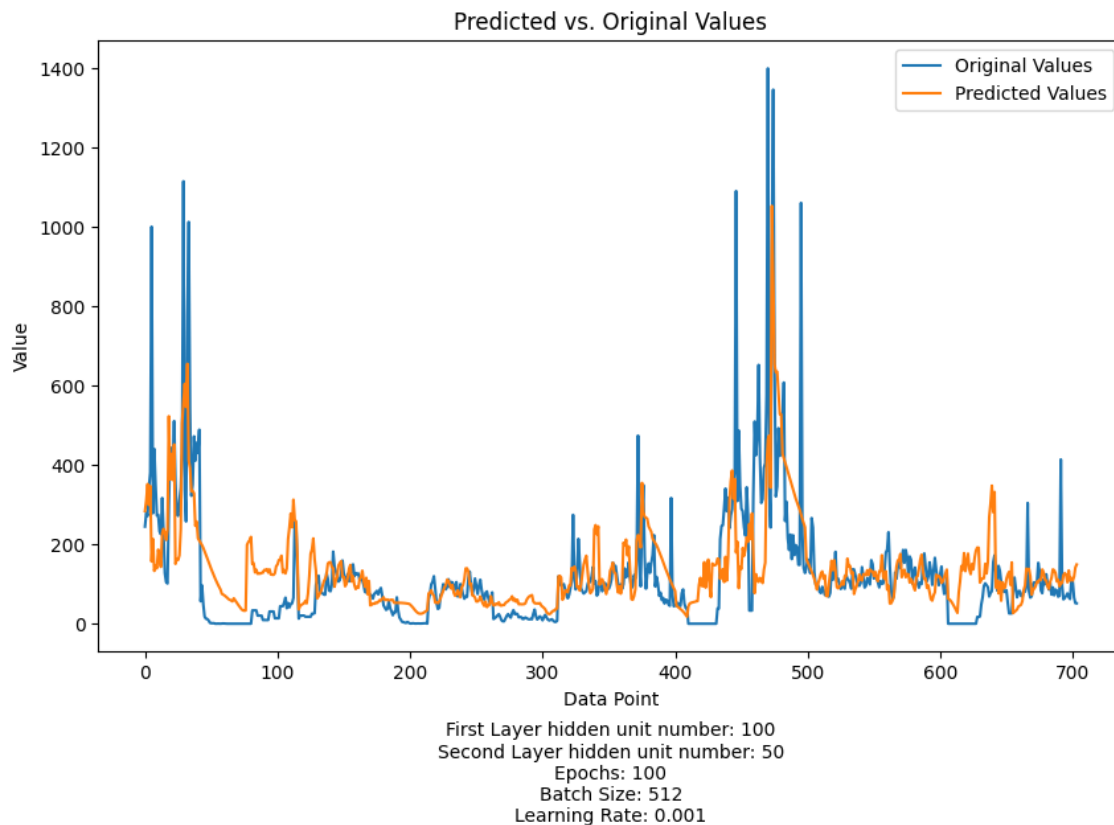
15. Ábra – Tesco eladások Kullback-Leibler [20] eloszlása Forrás: Saját szerkesztés

Ezután elemzésre került a kapott KL számok eloszlása és a következtetés az volt, hogy az eloszlás nagy mértékben hasonló (16. Ábra) és a sell out adatok generálása feltételezhetően elvégezhető a többi boltra levetítve is.



16. Ábra – Boltok Kullback-Leibler [20] normális eloszlása Forrás: Saját szerkesztés

Ezután a vizsgálat után, megtörtént a feltanítás és az előrejelzés egy Bidirectional LSTM modell használatával az előző dolgozatban taglalt módon, ami az alább látható tendenciát írta le. (17. Ábra)



17. Ábra - A prototípus szoftver valós és előrejelzett adatai egymáshoz viszonyítva Forrás: Saját szerkesztés

Az előrejelzett adatok kimerítő elemzése után megállapítható, hogy átlagosan 73,57 kg a tévedés 1 heti idő intervallum alatt a becsült és a valós adatok között, amely éves szinten akár a 3500 kg-os értéket is elérheti. Levonható a konklúzió, hogy a trendeket a szoftver kellően jól prognosztizálja, azonban a volumenbeli eltérések szignifikáns mértéket öltenek. Az előrejelzés minőségének javítása érdekében az adatok gazdasági környezetének diverzifikálása megfelelő megközelítés lehet. Ehhez külsős hírek kerülnek feldolgozásra, majd kiértékelésre.

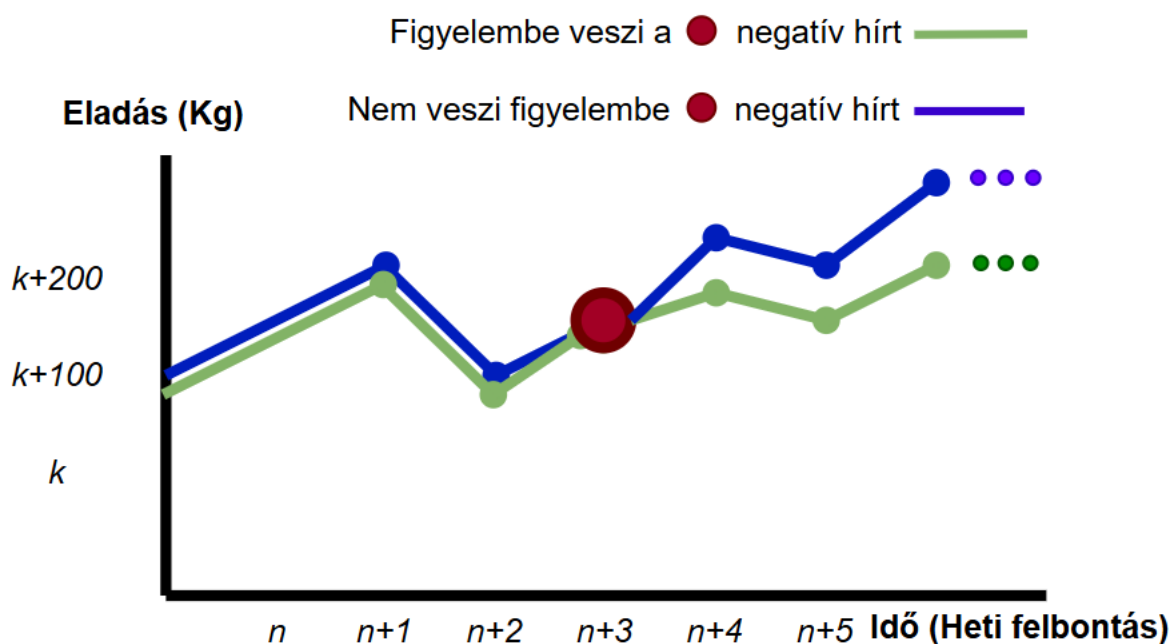
5. A hírfeldolgozáshoz fejlesztett szoftver

5.1 A hírfeldolgozás célja a kutatás folyamán

A hírfeldolgozás eleinte kizárólag azt a cél szolgálta, hogy elérhetővé váljon a kutatás számára egy módszer, ami számszerűsíthetően értékelhetővé teszi a média által lehozott cikkeket, ezáltal valamelyest betekintést nyújtson az az által képzett hatásra a fogyasztói társadalmon. Ennek a legmegfelelőbb módja a szentiment analízis végrehajtása volt a cikkek szövegtörzsén.

A média által lehozott hírek értékelésének automatizálása nehéz feladat az emberi nyelv sajátosságai miatt, mint a szarkazmus, többértelmű szavak, és jellemző az is, hogy egy-egy kijelentés nincs számokkal alátámasztva, de a fogyasztóközönségre mégis hathat. Ennek az akadálynak a leküzdésére egy LLM megfelelő lehet, hiszen párhuzamot képes vonni egy mondat értelme, és annak a gazdasági behatásai között, amit megfelelő utasítások hatására képes számszerűen értékelni.

Ez a számszerű érték egy további paramétert vezet be az LSTM modell tanítási fázisába, amely lehetőséget ad a modellnek nem csak a „mit”, hanem a „miértet” is megtanulni a váltakozó eladási adatok között. A következő példát tekintve (18. Ábra, Lásd: 32. oldal) eddig a modellnek rendelkezésre álltak termék specifikus értékek, illetve, hogy az adott héten volt e TV-s és internetes promóció. Ezekkel a második TDK-n bemutatott eredmények váltak elérhetővé. Megfigyelhető volt, hogy az előrejelzés volumene sokszor téves, azonban a trendek konzisztensen helyesek, tehát az esetek jelentős százalékában sikeresen előre jelezte az eladások csökkenését vagy növekedését, de a mértéküket nem feltétlen. A trendkövetési hibában a kiugrások magyarázhatóak azzal, hogy ugyan a modell tudja, hogy milyen számnak kell következnie a tanulásából kifolyólag, azonban megmagyarázni nem tudja a rendelkezésre álló paraméter mennyiséggel. Így, előfordulhatnak olyan előrejelzések, ahol a tanulásból kifolyólag egy növekedett eladás kellene következzen, azonban a megelőző héten egy, a nagy vállalat számára kiemelkedően előnytelen hírt hoz le több portál is, így az eladásai a várttól eltérőek, alacsonyabak lesznek.



18. Ábra, Az LSTM modell elvárt működése a hírek értékelésének figyelembevételével Forrás: Saját szerkesztés

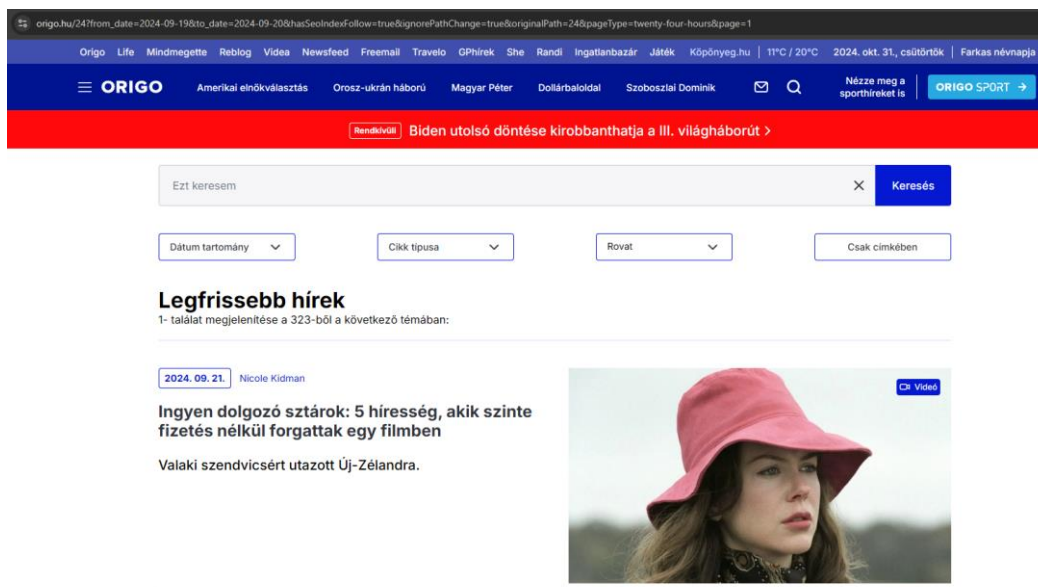
Ha az LSTM modell az előrejelzés pillanatában azonban megkapja az előző 2 hét paramétereit, és köztük szerepel egy paraméter, amely tartalmazza az előbb említett időtartam cikkeinek az összesített szentiment pontszámát, - a kutatás esetében ez gyakorlatilag egy piaci és gazdasági környezetet leíró pontszám a nagyvállalat szemszögéből- akkor lehetősége nyílik figyelembe venni, hogy az elmúlt időszakban a fogyasztókra negatívan ható hírek kerültek ki a médiába a cégről, így lehet, hogy a várttal ellentétben alacsonyabb lesz az eladások száma.

A nagyvállalat kapcsolattartójával való egyeztetés után az a megállapodás született, hogy mellett, hogy ez a rendszer egy kiegészítő eleme lesz az előrejelző rendszernek, érdemes lenne önmagában is elérhetővé tenni a nagyvállalat döntéshozó csapatának számára, mivel ők is fel tudják használni, azokat a kinyert, tömörített információkat, amiket a hírelemző rendszer közöl. Ahhoz, hogy a vállalat mindig tisztában legyen a portálok által lehozott cikkekel, önmagában is hatalmas mennyiségű cikk vizsgálatát igényelné (a kutatás folyamán vizsgált 5 portál heti szinten átlagosan 3500-4500 cikket közöl). Ezenkívül emberileg át kell olvasniuk a cikkek tartalmát, elemezni őket, és emellett meg is találni őket. A szoftver jelenleg képes heti szinten végig járni az 5 portál weboldalát, összegyűjteni, szűrni, értékelni majd pár mondatban összegezni a cikkek tartalmát, ezzel szignifikánsan redukálva az emberi erőforrás szükségletet, illetve jelentősen növelve az információ beáramlását a céges környezetbe.

5.2 Adatgyűjtés a hírportálokról

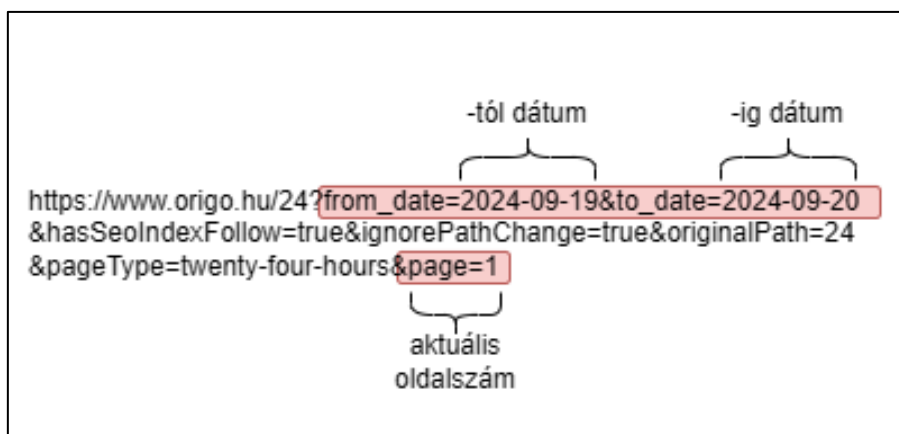
A hírek feldolgozása és kiértékelése egy sok lépéses folyamat, különösen, ha cél a nagyszámú források hatékony, szisztematikus kezelése. Az első lépés a hírek begyűjtése különböző oldalakról. Felmerült, hogy egy API használata lehetőséget adna erre, azonban ennek számos hibája lenne: korlátozta volna, hogy pontosan mely oldalakról történik a gyűjtés; azoknak mely adatait szolgáltatja, továbbá egy rendszeres passzív költséget vezet be a projektbe, illetve egy külső dependencia jelenne meg a munka folyamán. Ennek elkerülésére egy Python alapú hírgyűjtő szoftver került fejlesztésre, amely így teljes kontrollt biztosít a források kiválasztásában és a hírek közvetlen elérésében a kutatáshoz.

Első lépésként minden választott oldal forráskódját (origo.hu, femina.hu, index.hu, blikk.hu, vilaggazdasag.hu) elemezni kellett és megvizsgálni, hogy egy azonos struktúrában kinyerhetőek-e belőle a szükséges információk (metaadatok). Mivel minden oldal más megközelítést és megoldást igényelt, ezért egy általános leírás került az adatgyűjtés folyamatához (webscraping-hez). Először meg kellett arról bizonyosodni, hogy az adott oldalak rendelkeznek-e egy archív adatbázissal a híreikhez. (19. Ábra)



19. Ábra - Az origo.hu oldal archív adatbázisának keresője (Forrás: Origo.hu – Saját szerkesztés)

Amennyiben ez adott, és az URL-ből látszik, hogy a hírekre való tartomány kereshető és szűrhető, abban az esetben a Python kód alkalmazható a hírek letöltéséhez. Az URL manipulálásával látszik, hogy megadható egy -tól -ig tartományt, valamint az adott oldalt, ami tízesével kilistázza a híreket a tartományon belül. (20. Ábra)



20. Ábra - Az origo.hu oldal szűrőjének manipulálása Forrás: Saját szerkesztés

A program először végig iterál az összes oldalon, egészen addig amíg 404-es hibakódot nem küld az oldal. Minden oldalon a program a Python BeautifulSoup könyvtár segítségével a html tag-ek alapján beazonosítja a linkeket és strukturáltan egy fájlba lementi azokat a megadott dátum tartománnyal ellátva. (21. Ábra)

```
https://www.origo.hu/utazas/2022/11/hihetetlen-dolgot-muvelt-az-oroszlan-a-nevetgelo-turistakkal-video
https://www.origo.hu/sport/sport-futball/2022/11/juventus-inter-seria-a-lautaro-martinez-simone-inzaghi-juan-cuadrado
https://www.origo.hu/teve/2022/11/kiderult-hogy-van-most-a-pszichiatrian-kezelt-bb-evi-tv
https://www.origo.hu/nagyvilag/2022/11/gyanus-feher-port-kuldték-arizona-allam-republikanus-kormanyzojeloltjenek
https://www.origo.hu/itthon/2022/11/gazolt-a-vonat-budapesten-1
https://www.origo.hu/teve/2022/11/varatlan-vendeg-ebresztette-gaspar-zsoltit-video-tv
https://www.origo.hu/teve/2022/11/sztarban-sztar-leszek-otodik-elo-adas-2022
https://www.origo.hu/sport/sport-futball/2022/11/nb-i-fehervar-ferencvaros-osszefoglalo
https://www.origo.hu/teve-galeria/2022/11/a-sztarban-sztar-leszek-emlekezeses-pillanatai-kepek-tv
https://www.origo.hu/teve/2022/11/ujabb-bohockodas-az-rtl-hiradojaban-2022
https://www.origo.hu/sport/sport-csapat/2022/11/kezilabda-europa-bajnoksag-magyar-horvat-nyilatkozatok-vamos-petra-kuczora-csenge-tovizi
https://www.origo.hu/gazdasag/2022/11/sztar-hiresseg-ado-hatosag-tartozas
```

21. Ábra - A lementett cikkek linkjeinek összessége (részlet) Forrás: Saját szerkesztés

Miután a linkek összegyűjtésre kerültek, egy következő program blokk elkezd az linkekhez tartozó oldalakról kigyűjteni az adatokat egy JavaScript Object Notation (JSON) struktúrába. A kutatás számára releváns tartalmat az oldal html tagjei alapján lehet szűrni, így megtartható, vagy bizonyos esetekben eldobható némely rész az oldal tartalmából (például képek, vagy beágyazott reklámok). (22. Ábra, Lásd: 35. oldal)

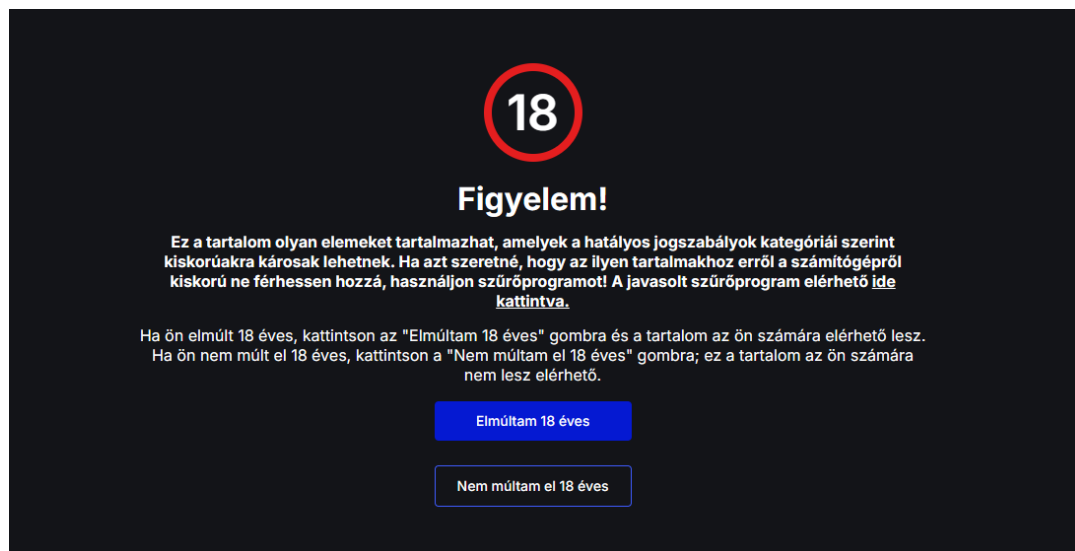
```

<meta name="keywords" content>
<meta name="og:title" property="og:title" content="Napokon belül itt van a 2023-as, androidos mobilok csúcsla
IGO">
<meta name="og:url" property="og:url" content="https://www.origo.hu/techbazis/2022/11/qualcomm-snapdragon-8-g
<meta name="og:image" property="og:image" content="https://cdn.origo.hu/2023/12/nW2wVm_ck4e0R46UxjVt4hyfEWbjo
oCJg/fill/1200/675/no/1/aHR0cHM6Ly9jbXNjZG4uYXBwLmNvbRlbnQucHJpdmF0ZS9jb250ZW50LzljNzc3ZTE0ZDkyYzRjODk4YmQ5N
0ZDg4.webp">
<meta name="og:image:width" property="og:image:width" content="1200">
<meta name="og:image:height" property="og:image:height" content="600">
<meta name="og:image:secure_url" property="og:image:secure_url" content="https://cdn.origo.hu/2023/12/nW2wVm_
jVt4hyfEWbjo-SKspK1bVroCJg/fill/1200/675/no/1/aHR0cHM6Ly9jbXNjZG4uYXBwLmNvbRlbnQucHJpdmF0ZS9jb250ZW50LzljNzc
zRjODk4YmQ5NmE1OTcxOTM0ZDg4.webp">
<meta name="og:type" property="og:type" content="article">
<meta name="og:description" property="og:description" content="Friss hírek Magyarországról és a nagyvilágból,
chnika, szórakozás, tudomány, kultúra. Az ORIGO a legfrissebb hazai és nemzetközi hírekkel szolgál olvasói sz
<meta name="og:locale" content="hu_HU">
<meta name="og:site_name" property="og:site_name" content="ORIGO">
<meta name="article:published_time" property="article:published_time" content="2022-11-06T18:45:00.000Z">

```

22. Ábra - Az oldal HTML kódja, ami tartalmazza a metaadatokat (részlet). Forrás: Origo.hu - Saját szerkesztés

Fontos megjegyezni, hogy megjelenhetnek felnőtt tartalomként megjelölt cikkek is az oldalon (például halálesettel kapcsolatos tartalmak), ezeket a program detektálja és egy Selenium nevű Python könyvtár egy virtuális chromium alapú chromedriver-t futtat, ami kikerüli és megnyomja az „Elmúltam 18 éves” gombot az oldalon, hogy folytatható legyen a tartalmak letöltése. (23. Ábra)



23. Ábra - Felnőtt tartalomként megjelölt cikkek nyitó oldala Forrás: Saját szerkesztés

A metaadatokat az előzetesen megállapított preferenciák szerint a program letölti, és egy standard JSON formátumra hozza. Lementésre kerül a cikk címe, leírása, a kulcsszavak, maga a típusa a cikknek, a tartalma, a publikálás dátuma, a nyitókép, az URL (a visszakereshetőség érdekében), valamint az ország, ahonnan származik a cikk (későbbi nemzetközi hírportálok

behozatalára tekintendő figyelemmel) és végezetül pedig maga az oldal amin megjelent a tartalom. (24. Ábra)

```
{
  "articleTitle": "Napokon belül itt van a 2023-as, androidos mobilok csúcslapkája",
  "articleDescription": "November közepén esedékes a Qualcomm Snapdragon Summit 2022, ezen debütál a Snapdragon 8 Gen 2",
  "articleKeywords": "Android, Snapdragon, Qualcomm, telefon, Teljesítmény",
  "articleType": "article",
  "articleContent": "A Qualcomm következő éves, nagy mobilos chipbemutatóját novemberre vártuk, most azonban már nem",
  "articlePublishDate": "2022-11-06T18:45:00.000Z",
  "articleURL": "https://www.origo.hu/techbazis/2022/11/qualcomm-snapdragon-8-gen-2",
  "articleThumbnailPicture": "https://cdn.origo.hu/2023/12/nW2wVm_ck4e0R46UxjVt4hyfEWbjo-SKspK1bVroCJg/fill/1200/675",
  "articleOriginatedCountryFrom": "hu_HU",
  "articleFromSite": "ORIGO"
}
```

24. Ábra - A strukturált JSON formátum, amibe a metaadatok kerültek Forrás: Saját szerkesztés

Majd miután végzett az adott oldalon a program, azt letárolja egy {from_date} – {to_date}_{articleNumber}.json fájlba. (25. Ábra)

Név	Módosítás dátuma	Típus	Méret
2022-10-31-2022-11-06.txt	2024. 09. 30. 17:45	Szöveges dokume...	99 KB
2022-10-31-2022-11-06_article_1.json	2024. 09. 30. 17:45	JSON File	2 KB
2022-10-31-2022-11-06_article_2.json	2024. 09. 30. 17:45	JSON File	5 KB
2022-10-31-2022-11-06_article_3.json	2024. 09. 30. 17:45	JSON File	2 KB
2022-10-31-2022-11-06_article_4.json	2024. 09. 30. 17:45	JSON File	3 KB
2022-10-31-2022-11-06_article_5.json	2024. 09. 30. 17:45	JSON File	1 KB
2022-10-31-2022-11-06_article_6.json	2024. 09. 30. 17:45	JSON File	2 KB
2022-10-31-2022-11-06_article_7.json	2024. 09. 30. 17:45	JSON File	2 KB
2022-10-31-2022-11-06_article_8.json	2024. 09. 30. 17:45	JSON File	4 KB
2022-10-31-2022-11-06_article_9.json	2024. 09. 30. 17:45	JSON File	1 KB

25. Ábra - A letöltött JSON fájlok összessége a program futását követően (részlet)Forrás: Saját szerkesztés

A megfigyelt időtartam 2022 október 31-től, 2023 november 26-ig terjedt ki (ez az időtartam egybevág a vállalati adatokkal). A fentebb említett oldalakról ilyen módon összesen 198.504 db cikk került letöltésre. Az oldalakon külön előszűrés nem történt a kulcsszavakra a releváns tartalmakkal kapcsolatban, abból a megfontolásból, hogy dinamikusan lehessen bővíteni a releváns keresési tartományt igény esetén (azért, hogy ne kelljen újra lefuttatni az összes oldalra a letöltéseket, csupán 1 megváltozott kulcsszóból fakadóan). Így a szűrésre a későbbiekben kerül sor az adatbázis feldolgozása folyamán, nem pedig a begyűjtés folyamán. Ezen kívül, így az adatbázis a továbbiakban alapul szolgálhat egyéb projektek folyamán is, nem specifikus a mostani kutatás témájához.

5.3 A begyűjtött adatok feldolgozása, adatbázisba rendezése

A webscraping folyamat után a következő szükséges lépés a feldolgozáshoz az adatok egységes formátumra hozása, illetve tárolása volt. Ennek több módja is elérhető volt, például egy Excel, .csv vagy .json fájlban, hiszen ezen formátumok módosíthatóak Python kódon keresztül, azonban kevesebb és rosszabb vizualizációt, illetve szűrést tettek lehetővé egy hagyományosabb adatbázishoz képest. Így a kutatás folyamán az SQLite adatbázis kezelő keretrendszer került kiválasztásra. Ehhez az adatbázishoz létezik Python könyvtár is, az sqlite3, amely még inkább megkönnyítette a kódon keresztüli manipulációt.

Az adatok az 5 különböző weboldalról 5 mappában voltak eltárolva, így egy Python script először végigjárta ezt az 5 mappát, mindegyikből készítve egy .sqlite adatbázist (26. Ábra).

blikk_hu	06/10/2024 02:12	File folder	
femina_hu	05/10/2024 16:36	File folder	
index_hu	08/10/2024 16:03	File folder	
origo_hu	15/10/2024 19:33	File folder	
vg_hu	14/10/2024 22:14	File folder	
articles_blikk_database.sqlite	08/10/2024 17:07	SQLite	156,552 KB
articles_femina_database.sqlite	08/10/2024 17:14	SQLite	42,376 KB
articles_index_database.sqlite	08/10/2024 16:42	SQLite	251,224 KB
articles_origo_database.sqlite	08/10/2024 15:58	SQLite	216,588 KB
articles_vg_database_base.sqlite	15/10/2024 18:06	SQLite	127,352 KB

26. Ábra A cikkek csoportosítása, a belőlük készült adatbázis Forrás: Saját szerkesztés

Mivel a korábban említett módon, a scrapelés folyamán azonos metaadatok kerültek letöltésre, az adatbázisok egybeolvasztása akadálymentes volt.

Ugyan mindegyik oszlop megegyező volt az oldalakról, az oszlopok egyedeinek (sorainak) formátuma eltérő lehetett. A cikk publikálásának ideje például mindegyik az ISO 8601 szabványnak megfelelő formátumban volt, azonban 3 portál a YYYY-MM-DDTHH:MM:SS±HH:MM formátumot alkalmazta, míg 2 a YYYY-MM-DDTHH:MM:SS.sssZ-t. Az első formátum az UTC+1 óra eltérést jelöli, míg az utóbbi az UTC-vel megegyező időformátum. A formátumbeli különbség leküzdésére egy új oszlop került hozzáadásra az adatbázishoz, az articleWeekOfYear. Ezért a feladatért egy Python script volt felelős, amely megvizsgálta egy egyed idő oszlopát, majd

meghatározta, hogy az évnek mely hetében publikálódott az a cikk. A kutatás folyamán azért esett a választás erre az egységes formátumra, mert ez összeillik a vállalat által szolgáltatott idő formátummal, hiszen ott is az eladások az év hetéhez vannak igazítva, nem pedig napi vagy havi lebontásban vannak.

A következő lépés az adatbázis szűrése volt, hiszen a begyűjtött cikkek jelentős része irreleváns volt a kutatás szempontjából. Ahhoz, hogy meglehessen állapítani, hogy egy cikk releváns -e, meglehetett vizsgálni a cikk címét, kulcs szavait, leírását és tartalmát. A legegyszerűbb módja ennek, ha egy kulcsszó felbukkan az előbb említett 4 oszlop valamelyikében, akkor az az egyed egy új adatbázisban mentésre kerül, mivel relevánsnak van bélyegezve. Ennek azonban vannak nyelvi akadályai, különösen a magyar nyelvben, hiszen némelyik szó tőhangváltáson esik át a ragozás folyamán, így lehet, hogy a cikk valamelyik mezőjében is csak ebben a formában fordul elő. Ilyen lehet például a „gabonapehely” szó, amely tárgyi ragot kapva „gabonapelyhet” szóvá alakul.

Ennek a problémának a leküzdésére a lemmatizációs (alapalakra hozás) eljárás nyújt lehetőséget. Ebben a folyamatban a program végig halad a megadott stringen, majd pedig visszaadja az összes szó alap alakját. (27. Ábra) Emögött a folyamat mögött egy nyelv specifikusan feltanított neurális háló áll. Ehhez, a Stanford University által kifejlesztett, és magyar nyelven is tanított, illetve elérhető modell áll, aminek a használatához a stanza Python könyvtár került felhasználásra.[21]

A vásárlók reggelire gyakran választanak különböző ízesítésű gabonapelyheket A vásárló reggeli gyakran választ különböző ízesítés gabonapehely
--

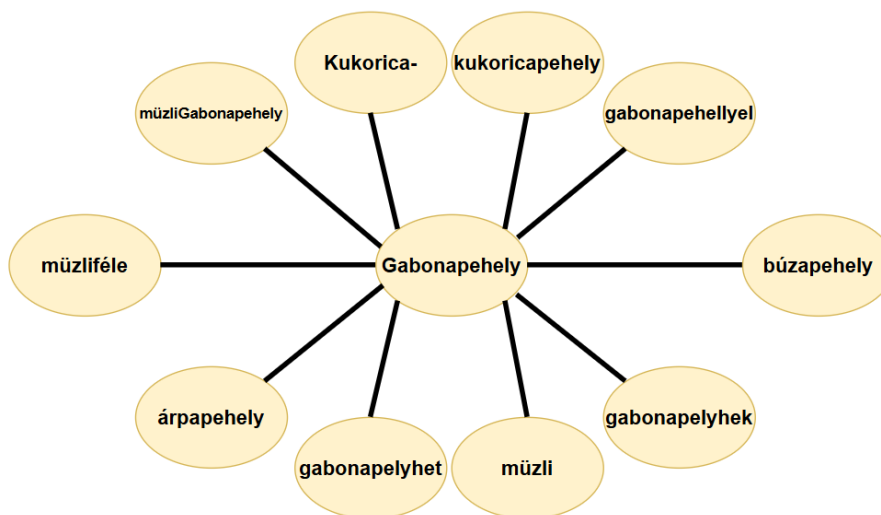
27. Ábra Lemmatizálás a gyakorlatban Forrás: Saját szerkesztés

Ennek a folyamatnak az eredményeként az adatbázis kibővült a lemmatizedKeywords, lemmatizedTitle, lemmatizedDescription és lemmatizedContent oszloppal, amely az ezekhez tartozó eredeti oszlop alapalakra hozott adatait tartalmazza.

A következő feladat az adatbázis feldolgozás folyamán a kulcsszavak meghatározása volt. Ez egy alapvetően manuális feladat, ahol intuitíven kell meghatározni, hogy mely szavak fontosak

a kutatás szempontjából. A teljes lefedettség érdekében azonban felmerült az ötlet, hogy kerüljön alkalmazásra egy vektor embedding modell, amely alkalmas magyar szöveg feldolgozásra. Ennek a célja az volt, hogy találjon olyan kulcsszavakat, amelyek logikai értelemben közel állnak azokhoz (tehát kicsi a távolság a vektorjaik között matematikai értelemben a vektor térben), amik intuitívan meghatározásra kerültek, azonban mégsem kerültek alkalmazásra emberi hiba (felejtés) folytán. Ehhez a Meta cég által fejlesztett [22], és magyar nyelvű modellal is elérhető FastText nevű vektor embedding modell család, illetve az általuk nyújtott fastText Python könyvtár került felhasználásra.

A célja az volt, hogy megadva például a „kakaó” szót a modellnek, az visszaadja a hozzá 10 legközelebb eső szót, majd, ha az a 10 szó tartalmaz olyat, amely intuitív módon hasznosnak bizonyulna, adódjon hozzá az eredeti kulcsszó halmazhoz. (28. Ábra)



28. Ábra A gabonapehely szóhoz legközelebb eső 10 szó (magát nem beleértve) Forrás: Saját szerkesztés

Mint az ábrán is látható, a megközelítés alapvetően helyes volt, azonban végül nem volt hasznos. Sokszor a 10 legközelebbi szó olyasmi volt, ami a lemmatizációs folyamat alatt mindenképpen a szótőre redukálódott volna, például a „gabonapehelyel”, „gabonapelyhet”, „gabonapelyhek”, stb. mind a „gabonapehely” szóra redukálódnak. Ez azért van, hiszen a 10 legközelebb eső találat a szóhoz rendszerint az alapszó valamilyen ragozott formája. Az angol nyelvben például ez egy sokkal megfelelőbb megközelítés lenne, mivel az előbbi 3 példát alapul véve, ezeknek a fordítása „with cereal”, „... cereal ...”, „cereals” lenne. Ez abból ered, hogy a magyar nyelv egy agglutináló (toldalékoló) nyelv, míg az angol egy analitikus (izoláló) nyelv, így ott az első 10 találatból átlagosan csak 2-3 találat esne bele ebbe a ragozott kategóriába. Az

indok amiért nem került növelésre a keresett szavak száma 10-ről például 20-ra, az az, hogy nem hozott javulást, mivel egyszerűen olyan szavakat talált a modell a vektor terében, amelyek nem léteznek. A cukor szó például számtalan „cukoré”, „cukor50”, „cukor.” találatot adott vissza. Ez valószínűleg abból ered, hogy a magyar nyelven kevesebb tanulás ment végbe a modellen, így nem dolgozott fel elég mennyiségű szöveget, hogy megfelelő logikai értékrend alakuljon ki benne. a szavak között. Továbbá, ha bármi olyan szó került vizsgálatra, ami valamilyen módon értelmezhető angolul is, például a „fitness”, akkor megfigyelhető volt, hogy szinte mind a 10 legközelebbi szó angol volt. Ennek a megközelítésnek így végül nem volt gyakorlati hozzáadott értéke, és nem került felhasználásra, így manuálisan épült fel a kulcsszó halmaz.

A projekt folyamán több kulcsszó halmaz került tesztelésre, kezdve a kevésbé szigorúakkal, haladva a szigorúbb felé. Ehhez a szűréshez a kulcsszó halmaz és az adatbázis lemmatized oszlopai kerültek összevetésre, ha valamelyik oszlop tartalmazza valamelyik kulcsszót, akkor az a teljes sor hozzáadásra kerül egy új adatbázishoz. Az első halmaz tartalmazott szinte minden olyan szót, ami elképzelhető volt, hogy relevánsa tesz egy cikket. (3. Táblázat)

Kulcsszavak száma	Talál cikkek	Talált releváns cikkek %-a
195	14597	0,07%
66	7561	0,04%
31	1115	<0,01%

3. Táblázat, A talált releváns cikkek száma, az alkalmazott kulcsszavak számával összevetve Forrás: Saját szerkesztés

A szűrés folyamán egy .txt fájlba végig logolásra (feljegyzésre) került, hogy melyik cikkek kerülnek be a szűrt adatbázisba (/Ábra/ egy -txt-ből). Ehhez az egyedi ID-ja került lementésre, illetve mellé vesszővel elválasztva, hogy melyik kulcsszó (/kulcsszavak) miatt került kiválasztásra. Az első szűrésből megállapítható volt, hogy az olyan szavak, mint a „cukor”, „kakaó”, „csokoládé”, „egészség”, „reggeli” stb. túl generikusak, és rengeteg olyan cikk kerül beválogatásra, amelyek nem relevánsak. További akadály volt még, hogy a közmédiában a „csoki” például egy személy beceneve, vagy a vállalat némelyik termékének neve megegyezik egy híres sportoló nevével, ami által rengeteg nem megfelelő cikk került beválogatásra. (29. Ábra)

```
A 81. ID-jú cikk áthelyezésre került a filteredMergedDataBase.sqlite fájlba a következő kulcsszó miatt: gabonapehely
A 312. ID-jú cikk áthelyezésre került a filteredMergedDataBase.sqlite fájlba a következő kulcsszó miatt: kakaó
A 340. ID-jú cikk áthelyezésre került a filteredMergedDataBase.sqlite fájlba a következő kulcsszó miatt: gabonapehely
A 789. ID-jú cikk áthelyezésre került a filteredMergedDataBase.sqlite fájlba a következő kulcsszó miatt: <cég_neve>
A 1881. ID-jú cikk áthelyezésre került a filteredMergedDataBase.sqlite fájlba a következő kulcsszó miatt: reggeli
A 2042. ID-jú cikk áthelyezésre került a filteredMergedDataBase.sqlite fájlba a következő kulcsszó miatt: tej
A 3149. ID-jú cikk áthelyezésre került a filteredMergedDataBase.sqlite fájlba a következő kulcsszó miatt: <cég_terméke>
```

29. Ábra A logolási folyamata az adatbázis szűrésének Forrás: Saját szerkesztés

Felmerülhet a kérdés, hogy miért kerültek begyűjtésre bulvár weboldalak, miért nem csak gazdasági weboldalak kerültek feldolgozásra, hiszen azok számottevően több, és relevánsabb információt hozhatnak le egy multi cég piaci képéről. Ennek az oka, hogy a fogyasztói társadalom jelentős része nem gazdasági portálok tartalmát fogyasztja, hanem bulvár lapokét. Így hiába is célja a rendszernek előre jelezni az eladásokat a gazdasági portálok által nyújtott piaci kép alapján, ha ez a piaci kép nem jut el a fogyasztói társadalomhoz és nem befolyásolja őket. Ebből a megfontolásból, érdemes lehet a forrásokat megkülönböztetni a jövőben „bulvár” és „gazdasági” jelzővel, így differenciálva a fogyasztókhöz eljuttatott információt (bulvár) és a valós piaci képet leíró információt (gazdasági). Ez a megközelítés azonban a kutatás jövőbeli részét képezi.

A kutatás végül a legszigorúbb kulcsszó halmazt alkalmazta, amely csupán 31 elemből állt. Ezek mind olyan szavak, amik kellően cég specifikusak, csak az ő termékeiket tartalmazza, és azok közül is csak azokat, amik nem vonzanak be akár több száz irreleváns cikket is, így végül 1115 darab cikk maradt a releváns halmazban. Fontos megjegyezni, hogy ezek azok a hírek, amelyek fontosnak ítéldtek meg a kulcsszavak alapján, de nem biztos, hogy tényleg relevánsak.

5.4 Szentiment analízis LLM felhasználásával

A szentiment analízis végrehajtásához egy LLM alapú megközelítés került kiválasztásra. Ennek az oka, hogy ez nyújtja a legintelligensebb értékelést amikor hosszabb, összefüggő szöveg értékelésre van szükség, mint például a cikkek szövegtörzse. Ezen kívül, a nyelvi modellek képesek kontextualizálni az elemzést, tehát nem csak megállapítja, hogy pozitív vagy negatív a hír, hanem azt bizonyos szempontokból képes elemezni is. Lehetőséget ad arra is, hogy az elemzés mellé ne csak egy pontszám társuljon, mint ahogy az egy szabályrendszer vagy szimp-lán gépi tanulós módszerrel történne, hanem egy indoklást is. Ezenkívül pedig képes tovább szűrni az adatbázist, hogy kizárólag olyan hírek befolyásolják később az LSTM modell működését, amelyek valóban relevánsak, illetve, hogy a vállalathoz is csak olyan cikkek jussanak el az eszközükön keresztül, ami hasznos számukra.

A kutatás következő lépése, hogy kiválasztásra kerüljön a használandó LLM. Először a lokálisan futtatott verzió került kiértékelésre. A dolgozat készülése idején a projekt nem rendelkezik kellő erőforrásokkal ahhoz, hogy egy saját, teljes mértékben pénzügyileg és erőforrás ügyileg

önállóan finanszírozzon egy LLM működést. Így a következő opció egy, a felhőben futtatott virtuális gép lenne kellő erőforrással mögötte, amely lehetővé teszi már egy jobb LLM futtatását kellő sebességgel.

Itt felmerülhetnek biztonsági kérdések, mint például, hogy nem -e veszélyes a projektben lévő adatokat a felhőbe küldeni. Hosszas elemzés után arra a konszenzusra jutottunk, hogy a projekt ezen része folytatódhat felhős megoldásokkal, mivel semmilyen céges adat nem kerül ki az internetre, ezáltal semmilyen titoktartási szerződést, se digitális biztonságot nem sért és nem veszélyeztet. A felhőbe kizárólag egy system prompt, ami a kutatók megfogalmazásából eredő szöveg; a weboldalakról begyűjtött cikk szövegtörzse és a cikkhez tartozó egyedileg azonosító cikk ID-ja kerül ki.

Először az Amazon Web Services (AWS), majd a Microsoft Azure által szolgáltatott virtuális gépek irányába indult a kutatás, azonban ezeket a gépeket megfelelően létrehozni majd üzemeltetni mind idő, pénz szempontból költségesebb, mint a harmadik megoldás, továbbá ezeknek az elkészítése önmagában felölel egy fél TDK dolgozatot.

Ezeket a kritériumokat és költségeket mérlegelve az OpenAI szolgáltatásai kerültek kiválasztásra. Több okból is ideális választás a projekt számára: pay-as-you-go (fizess, ahogy használsz) modellt alkalmaz, ami lehetővé teszi, hogy ne kelljen többet költeni annál, mint amennyi valóban felhasználásra került; state-of-the-art modell családdal rendelkezik a szolgáltatás, ami könnyen lehetővé teszi a váltást, amennyiben egy intelligensebb/olcsóbb/alkalmasabb modell válik elérhetővé; jelentősen olcsóbb bármilyen kompetenciánál a „chatgpt 4o-mini” modell. Ennél a modell használatnál 1 millió input token 0.15\$, míg 1 millió output token 0.65\$. Amennyiben batch (csoportos) feldolgozást indít a felhasználó, ez a 2 ár a felére csökken. Ez azt jelenti, hogy nem azonnal kap eredményt az API híváson keresztül, hanem 24 órán belül, mivel ezek a kérések akkor kerülnek feldolgozásra, amikor alacsonyabb a szolgáltatás kihasználtsága. A projekt jellegéből kifolyólag, a lekérésre a válasz nem azonnal szükséges, hanem csak egy bizonyos időn belül, így ez lehetővé teszi a költségek felezését.

Az API endpoint (végpont) az OpenAI oldalán egy JavaScript Object Notation Lines (jsonl) fájlt vár. Ez a fájl típus megegyezik a hagyományos .json kiterjesztésű fájljal, annyi különbséggel, hogy tulajdonképp minden sora 1 json fájlt tartalmaz. Egy Python script létrehozza ezt a .jsonl fájlt, majd feltölti a szűrt adatbázis sorainak a releváns mezőivel, ez az ID-je a cikknek

és a tartalma. Így összesen 1115 sor kerül ebbe a .jsonl fájlba. Ezután a soron belül meg kell határozni, hogy mely endpoint-ra megy a kérés, ami a „/v1/chat/completions”, a hívás metódusát, ami a kutatás esetében „POST”, a felhasználó által használni kívánt modell, ami a „gpt-4o-mini”.

A fentebb említett paramétereken túl a json jelentős eleme még a „messages” mező. Itt határozható meg a „system” és „user” role és a bennük/általuk közölt prompt. A system prompt lehetőséget ad arra, hogy meghatározza a felhasználó az LLM viselkedését, tájékoztathatja arról, hogy éppen kinek válaszol, milyen formátumban várja el a választ a felhasználó, miket vegyen figyelembe stb. A system promptban közölve van az LLM-el, hogy: Egy szentiment analízist kell végrehajtania, a <cég_neve> szempontjából, ahol az 1-es pontszám negatív, a 2-es semleget, a 3-as pedig pozitív hírt képvisel; json formátumban adja vissza a választ, ahol az első elem a cikk relevanciáját jelenti, 0-s értéket vegyen fel, ha a cikk mégsem releváns a céghez, 1-est ha mégis releváns, a második érték a json-ben az LLM által kiosztott 1-2-3 pontszám vagy -1, ha nem releváns a cikk, a harmadik érték pedig legyen egy szöveges összefoglaló arról hogy miről szólt a cikk, továbbá miért releváns agy irreleváns a cég számára.

Ezután a user prompt tartalmazza az adatbázisból beillesztett cikk tartalom mezőjét, ezt dolgozza fel az LLM.

További megadható paraméter, hogy maximálisan mennyi tokenben válaszoljon a rendszer, milyen formátumban tegye ezt, itt a „json” formátum került kiválasztásra, mivel alapvetően folyószöveget ad vissza, továbbá a „temperature” (személyiség) paraméter beállítása, ami egy 0-1 közötti szám, és a nyelvi modell kreativitását befolyásolja. Mivel a cél, hogy a projekt folyamán konzisztens elemzések készüljenek, hogy az LSTM modellt egyenlő mértékben befolyásolja 2 egyenlő mértékben rossz hír, így egy alacsonyabb temperature érték került kiválasztásra, hogy a válaszok minél kevésbé legyenek különböző skálán értékelve.

A batch .jsonl összeállítása és elküldése után a válasz hasonlóan ebben a formátumban érkezik vissza, ahol minden sor 1 json-t tartalmaz, és az a json tartalmazza az egyedi kérés összes adatát. A formátum helyes, konzisztensen egy json objektum volt a válasz, aminek az első eleme mindig 1 vagy 0, a második elem mindig egy pontszám, a harmadik eleme pedig a szöveges összegzés.

Az alábbi példán (30. Ábra) látszik, hogy a cikket relevánsnak ítélte meg az LLM, és mivel a hír olyan dolgokat tárgyal, amelyek pozitívak a fogyasztóközönség nagyobbik részében (a cikk arról szól, hogy a cég támogatja a kakaóbab beszerzésének és termesztésének etikus módját és lépéseket tesz ennek érdekében), így 3-as, pozitív szentiment pontszámot kapott. Az összefoglaló részben pedig pár mondatban összefoglalja a cikk szövegtörzsében tárgyaltakat.

```
{
  "relevancy": 1,
  "score": 3,
  "summary": "A cikk releváns a <cég_neve> gabonapehely fogyasztói közössége számára,
    mivel a kakaóbab beszerzésének etikai és minőségi aspektusai közvetlenül
    befolyásolják a csokoládé és gabonapehely termékek minőségét.
    A cikk hangsúlyozza az etikus beszerzés fontosságát,
    amely a fogyasztók számára egyre fontosabbá válik,
    így a pozitív szentiment pontszám is indokolt."
}
```

30. Ábra Példa, ami rámutat a releváns cikkek helyes kezelésére Forrás: Saját szerkesztés

A következő példa (31. Ábra) bemutatja, hogy milyen, amikor egy olyan cikk is tovább jut a kulcsszó halmaz általi szűrésen, ami teljes mértékben irreleváns. Látható, hogy a rendszer irrelevánsnak ítélte meg. Mivel irreleváns a cikk, a score-ja -1, ami annyit jelent, hogy nem kerül értékelésre. Ezeket a cikkeket nincs értelme így értékelni, mivel lehet, hogy egy másik projekt keretein belül ez a cikk is fel lesz dolgozva, azonban ahhoz egy új system promptra van szükség. Így, ezek a cikkek nem kerülnek kiértékelésre. Továbbá a végén olvasható az összefoglaló arról, hogy miért került irreleváns megítélésre a cikk.

```
{
  "relevancy": 0,
  "score": -1,
  "summary": "A cikk a Lidl játék konyhájának népszerűségéről és áráról szól,
    ami nem kapcsolódik közvetlenül a <cég_neve>
    gabonapehely fogyasztói közösségéhez.
    A cikk tartalma nem érinti a <cég_neve> termékeit vagy azok fogyasztói szokásait,
    így nem releváns a megadott szempontból."
}
```

31. Ábra Példa, ami rámutat az irreleváns cikkek helyes kezelésére Forrás: Saját szerkesztés

Vegyünk egy utolsó példát (32. Ábra, Lásd: 45. oldal), amely bemutatja az LLM-ek hibáját, a hallucinációt. A hallucináció az LLM-ek esetében olyan jelenség, amikor a modell magabiztosan generál valótlan, helytelen vagy kitalált információkat. Ez akkor fordul elő, amikor a modell a bemenetek alapján olyan választ alkot, amely nem tükrözi a tényleges valóságot vagy nem

támasztható alá hiteles forrásokkal, hanem pusztán a statisztikai mintázatok alapján formálódik ki a tokenek generálása folyamán. Látható, hogy a modell helyesen megállapítja a relevanciáját a cikknek, azonban utána magabiztosan mond ellent magának: a szentiment pontszáma 2, tehát semleges. Azonban a magyarázatában alátámasztja, hogy a történések inkább negatív hatással lehetnek a fogyasztói közösségre, mint semleges vagy pozitív, majd ezután kijelentést tesz, hogy semleges a cikk szentimentje. Az indoklása lehetne helyes is, miszerint a búza árának csökkenése egy pozitív dolog, míg a kávé árának emelkedése negatív dolog, így a cikk összességében semleges, azonban nem ezt teszi.

```
{
  "relevancy": 1,
  "score": 2,
  "summary": "A cikk a mezőgazdasági termékek áralakulásáról és a gabonapiac helyzetéről szól,
    ami közvetlenül érinti a <cég_neve> gabonapehely fogyasztói közösségét.
    A búza árának csökkenése és a kávé árának drámai esése negatív hatással
    lehet a gabonapehely termékek árára és elérhetőségére,
    így a cikk semleges szentimentet kapott."
}
```

32. Ábra Példa, ami rámutat a modell eseti hibás pontozására / indoklására Forrás: Saját szerkesztés

5.5 RAG architektúrára épülő szentiment analízis

A RAG architektúra alkalmazásának célja az LLM-ek általi hallucinációk csökkentése, illetve az általános kiértékelés minőségének növelése. Ahhoz, hogy meglehessen állapítani, hogy ez a változtatás valóban sikeres volt, érdemes egy rossz hírfeldolgozást alapul venni, majd összevetni a RAG rendszer által nyújtott értékelés eredményével. (33. Ábra)

```
{
  "relevancy": 1,
  "score": 2,
  "summary": "A cikk releváns a <cég_neve> gabonapehely fogyasztói közössége szempontjából,
    mivel az árstop intézkedések hatással lehetnek az élelmiszerárakra,
    beleértve a gabonapelyheket is. Az inflációs nyomás és az alapvető élelmiszerek
    árának befagyasztása miatt a fogyasztók érzékenyebbek lehetnek az árak alakulására,
    ami befolyásolhatja a <cég_neve> termékeinek keresletét."
}
```

33. Ábra Példa, ami bemutatja a hallucinálást

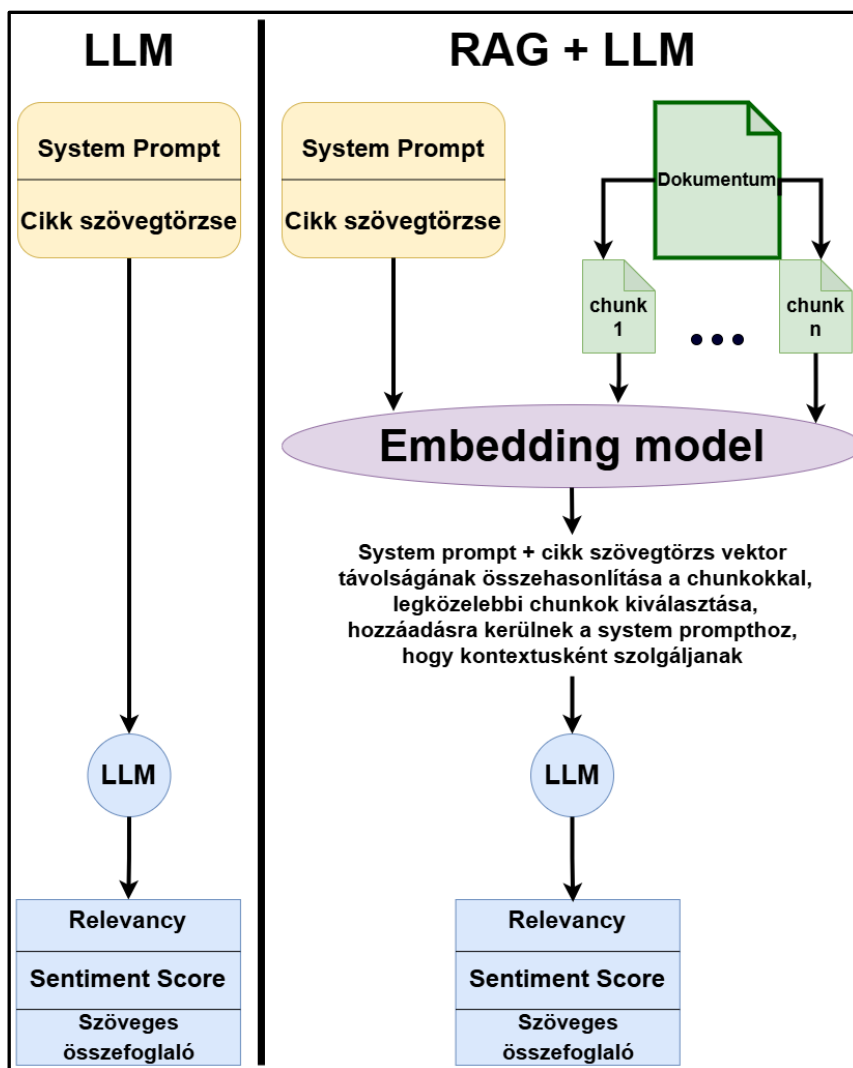
Annak ellenére, hogy a cikk összefoglalója azt tartalmazza, hogy az intézkedések nyomást helyeznek a vásárlóra és érzékenyebbé teszik pénz költés terén, ami direkt módon befolyásolhatja a termékek keresletét, a szentiment pontszáma mégis 2, pedig logikusan 1-nek kellene lennie.

Ahhoz, hogy az LLM hozzáférhessen a cikkhez specifikus tudáshoz, először azt vektorizálni kell. Erre a kutatás folyamán a [23]World Bank által kibocsátott kutatás dokumentum szolgál alapul, amely az árstop negatív hatásait vizsgálja fejlett és fejlődő országok között.

Először ez a .pdf dokumentum a Python PyMuPDF (Fitz) könyvtár segítségével megfelelő feldolgozásra kerül. Ehhez először oldalanként kinyerésre kerül a szöveg, amelyet ezután oldalanként sorba rendezünk. Ezután jönnek az olyan eljárások, ami a szöveg megtisztításáért felelős: üres karakterek eltávolítása, pdf formátumból eredő sorvégek összefűzése. A szöveg feldarabolására kisebb részekre ugyan nincs szükség, mivel az OpenAI által szolgáltatott „text-embedding-3-small” modell maximálisan 8191 token inputot képes befogadni egyszerre, és a kutatás szövege csupán ~7900 token volt (ennek a megállapításához az OpenAI tokenizer webes felülete volt alkalmazva) [24], azonban érdemes logikai/daraboló egységekre (chunkokra) felbontani a dokumentumot. A méretüknek 200 tokent került beállításra, 50-es átfedés értékkel. Ez azt jelenti, hogy egy chunk maximum 200 token, és ezek tartalmazzák az őket megelőző, illetve utánuk következő 50 tokent, azért, hogy a szöveg ne kerüljön teljesen összefüggéstelenné. Tehát az első chunk 0-200-ig tartalmazta a tokeneket, a második 150-350-ig, és így tovább.

Az OpenAI API endpointja az embedding modellek esetén v1/embeddings, ami POST method-al érhető el. Bemeneti paraméterei az „input”, ez a vektorizálandó szöveg, a kutatás esetében az előállított .txt fájl a dokumentumból, illetve a „model” paraméter, ami meghatározza a modell választást, a projektben ez a fentebb említett „text-embedding-3-small”.

A PDF vektorizáció után a következő lépés a cikk szövegtörzsének a vektorizációja, ugyanis ezt a vektort kell hasonlóság keresésnek alávetni. Miután mind a dokumentum, mind pedig a cikk szövegtörzse vektorizálva lett, a numpy Python könyvtár segítségével meghatározható, hogy mely dokumentum részletek (chunkok) a logikailag legközelebb állók a cikk szövegtörzséhez. A cikk szövegtörzsével együtt a 3 legközelebb álló chunk került elküldésre ezután az LLM-nek. A system prompt kiegészítésre került azzal az információval a modell számára, hogy ezeket a forrásokat használja fel a szentiment analízishez. (34. Ábra, Lásd: 47. oldal)



34. Ábra A RAG rendszer folyamatábrája összevetve a csupán LLM-et használóval Forrás: Saját szerkesztés

A cikk releváns részleteit felhasználva, a modell a következő szentiment analízis eredményt adta: (35. Ábra) (A szöveg kutatók általi fordításon esett át, mivel az angol kontextus hatására, angolul válaszolt a modell.)

```
{
  "relevancy": 1,
  "score": 1,
  "summary": "A cikk releváns a <cég_neve> gabonapehely fogyasztó közösség számára. Az árstop intézkedések a források alapján rendszerint negatív hatással vannak a fogyasztók vásárlási kedvére, mivel a gazdaság kollektív belassulását okozza, így az FMCG cikkek mozgása is valószínűleg lecsökken, ami negatívan befolyásolhatja a <cég_neve> eladásait."
}
```

35. Ábra Példa, ami bemutatja a RAG által javított korábbi példát Forrás: Saját szerkesztés

Mint látható, a modell alkalmazta a kontextust, amit kapott, és ennek hatására új pontszámot kapott a cikk, most már logikailag helyeset. Ezenkívül, olyan tudást is belevitt a válaszába (FMCG termékek karakterisztikái árstop esetén) amik lehet, hogy ugyan megtalálhatóak voltak a tanulási súlyaiban is, azonban ezeket nem alkalmazta.

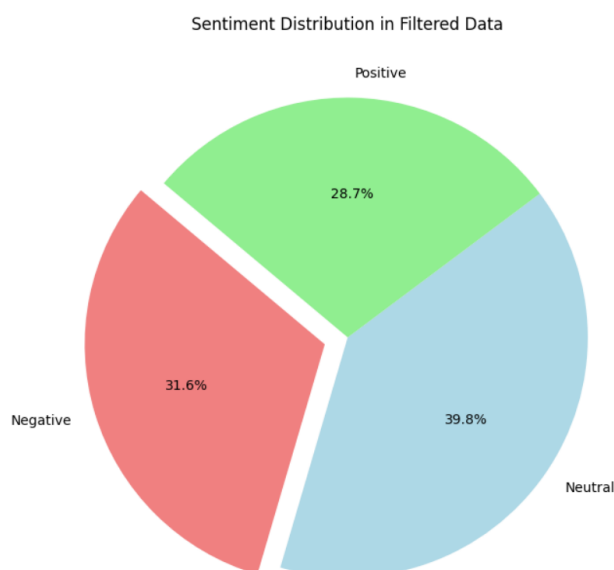
6. Az eredmények

Egy Python script segítségével kiszűrésre kerültek azok a json-ok, amelyek releváns cikk értékelést tartalmaztak. Az kezdeti 198.504 cikkből 342 darab cikk maradt végül, amely mind kulcsszavak alapján, mind pedig az LLM logikai értékelése alapján is relevánsnak minősültek. Ez azt jelenti, hogy átlagosan napi ~1 cikket írnak, ami releváns a cég számára, ezen az 5 portálon, ebben az 1 éves időintervallumban. (4. Táblázat)

Negatív	Semleges	Pozitív
108	136	98

4. Táblázat A hírek eloszlása kategóriánként. Forrás: Saját szerkesztés

Kimutatható, hogy a rendszer elemzése alapján, ebben az időintervallumban a hírek nagyobbik többsége semleges, körülbelül ~40%, míg a pozitív és negatív hírek eloszlása közel megegyező, ~30-30% (36. Ábra)



36. Ábra A hírek százalékos eloszlása kategóriánként

A kutatás bebizonyította, hogy lehetséges a hírek szisztematikus begyűjtése az online portálokról szolgáltatott API nélkül is, ami lehetővé teszi a jövőben ennek az automatizálását, továbbá jelentős költségcsökkentést jelent a projekt huzamos finanszírozásának fényében.

Látható, hogy a megközelítés, miszerint portálokról kerülnek hírek begyűjtésre majd elemzésre véghez vihető, ugyanis elég cikk készül a téma relevanciájában, ami a számokat tekintve azt jelenti, hogy átlagosan heti ~7 cikk értékelése befolyásolhatja az LSTM modell előrejelzését. Fontos kiemelni, hogy ez kizárólag ezt az 5 magyar hírportált foglalja magában, azonban ez az adatbázis a projekt előrehaladtával növelhető, így a statisztikai szignifikanciája a hetekre lebontott piaci elemzésnek tovább növelhető.

A hírek szűrése lehetséges az alkalmazott módszerrel, tehát először kulcsszavak alapján szűrni őket, hogy a mennyiségük csökkentésre kerüljön, ami által a későbbi pénz, - idő és erőforrás használat is redukálható. A szűrés második fázisából, tehát az LLM logikai szűrése után látható, hogy 1115 cikkből végül csak 342 maradt végleg releváns, így ez egy további jelentős szűrési feltétel, hogy tiszta és hasznos adatbázis maradjon fent a szentiment analízishez.

A RAG metodológia helyesnek bizonyult a megvizsgált esetben, ami indikálja a további fejlesztéseket ezen a területen. Csökkent a hallucináció mértéke, mivel a modell válasza jobban korlátok közé került, a kontextusbeli szöveg alkalmazása miatt.

Az OpenAI szolgáltatásai a kutatás folyamán megfelelőnek bizonyultak, modern és könnyen integrálható state-of-the-art megoldásokat kínálnak, a kutatás költségének megfelelő kategóriában. A felhő szolgáltatás alkalmazása végül nem ütközött biztonsági / jogi akadályba, így a továbbiakban, amíg céges adat nem kell a felhőbe kerülni, ez a megközelítés megfelelőnek bizonyult.

Az elkészült szoftver keretrendszer elsődlegesen alkalmas a cégnek az aktuális hírek feldolgozására, ezzel segítve a munkájukat, másodlagosan pedig visszatekinteni a múltban lehozott hírekre, és azoknak a korrelációjára az eladási adatokkal. Ezen kívül, lehetővé teszi az LSTM modell finomhangolt tanítását, alkalmazva a hírek értékeinek összesítését, így megmagyarázva a „miértet” a „mit” mögött, ezáltal potenciálisan javítva nem csak az eladási trendek megfelelő előrejelzését, de a prediktált eladási volument is, ezzel növelve a pontosságát a forecastnak.

7. Konklúzió

Mivel a piac hektikus változásainak mértéke várhatóan nem fog csökkenni, sőt, inkább ezen a szinten marad, vagy nőni fog, így a vállalat példáján keresztül kijelenthető, hogy egy alternatív terveket engedélyező stratégia megoldást jelenthet a bevezetésben ismertetett gazdasági problémára.

Az első TDK alatt elért eredmények alátámasztják azt az elképzelést, hogy egy gépi tanulásos módszerekkel felépített neurális hálózat alapú rendszer képes kellően jó hibahatáron belüli eredményeket nyújtani, a hagyományos tervezéshez képest relatív 0 emberi idő és erőforrás befektetésével a vállalat szempontjából.

Az második dolgozat keretein belül az adatfeldolgozás, adattisztítás, illetve a hiányos adatok pótlása volt az elsődleges prioritás, míg a másodlagos prioritás egy új megközelítés kutatása volt a természetes nyelvi feldolgozásos rendszer elméleti működése által, ami lehetővé teszi a múltbeli eladási adatokon kívül egyéb fontos külső tényezők, mint a hírek, cég specifikus információk, globális adatok felhasználása az előrejelzéshez.

Az idei, tehát 2024 őszi félévében zajló kutatás az előző elméleti alapjaira építkezve megvalósította a hír gyűjtés és elemzés folyamatát, ami önmagában is a fejlesztett szoftver keretrendszer szerves része, azonban egyúttal egy proof-of-concept-ként (megvalósíthatóság bizonyítéka) is szolgál. A cég visszajelzései alapján kijelenthető, hogy ez egy hasznos fejlesztés volt, ami nem csak elméleti, hanem gyakorlati szinten is alkalmazható.

7.1 Jövőbeli munka

A kutatás során fejlesztett szoftver együttes már jelentős optimalizálásokon esett át, azonban még számos lehetőség áll rendelkezésre a projekt hír kiértékelő rész rendszer további fejlesztésére.

Elsődlegesen, érdemes tovább bővíteni a scrapelt oldalak számát. Minél nagyobb cikk halmazt van lehetősége az LLM-nek szentiment analízis alá vetni, annál elhanyagolhatóbbak azok az elemzések, amelyek hallucinációval vagy rosszul megítélt pontszámmal végződtek, a nagyszámok törvénye által.

Érdemes lehet megkülönböztetni a cikkeket abból a szempontból, hogy azok egy bulvár vagy egy valamivel formálisabb, gazdasági portálról származnak. Ezérdekében célszerű lehet korreláció vizsgálatot végezni, a külön bontott oszlop értékek (tehát, a bulvár lapokból összesített heti piaci kép, és a gazdasági lapokból összesített heti piaci kép) között.

A modell jelenleg a cikkekről kizárólag azt dönti el, hogy negatív, semleges vagy pozitív egy hír értéke. Érdemes azonban megjegyezni, hogy attól, hogy két különböző hír is a negatív kategóriába esik, nem feltétlen egyen értékűek, hiszen nem mindegy, hogy a tej drágulása negatívan befolyásolhatja a müzli fogyasztás mértékét Magyarországon, vagy mondjuk gyomorrontást okoznak a cég bizonyos termékei. Így, egy szélesebb skála, ami nem csak azt határozza meg hogy valami negatív vagy pozitív, hanem azt is, hogy mennyire negatív vagy pozitív, jó irányba tereli a szentiment analízis minőségét.

A cikkek szűrésére a kutatásban kipróbált mód gyors, és költség hatékony. Fennáll az esélye azonban, hogy vannak olyan cikkek, amelyek logikailag relevánsak a vállalat számára, mégsem kerültek be a kulcsszó halmaz szűrése alapján. Mind a ~200.000 cikk LLM-el való szűrése nem költséghatékony folyamat, azonban lehet, hogy növeli a releváns cikkek számát, úgy, hogy az nem megy az adatbázis minőségének rovására. Így, a jövőben érdemes szem előtt tartani ezt a lehetőséget a cikkek számának bővítésére.

Az elmúlt évet figyelembe véve kijelenthető, hogy szinte minden hónapban készült egy új LLM modell, amely felülmúlja az előző state-of-the-art modellek benchmark eredményeit. Nem garantált, de realisztikusan arra számíthatunk, hogy a jövőben is további fejlődéseken mennek át ezek a modellek, amelyek csökkenthetik a hallucináció esélyét, illetve növelhetik az általános logikai készségeiket. Így a modell lecserélése egy jobban teljesítőre szinte mindig egy egyértelmű lépés a minőségi javulás irányába.

Mivel a RAG rendszer prototípusa az elvárt javulásokat hozta, így érdemes ezt a metodológiát széleskörűbben is alkalmazni az elemzések folyamán. Ehhez szükséges egy vektor adatbázis létrehozása és használata, egy szimpla memóriában tárolt vektor embedding helyett. Ezt az adatbázist a korábbi példában említett mellett egyéb, gazdasági anomáliákat tárgyaló, fogyasztói közösség és média kapcsolatát vizsgáló, FMCG termékekre kiterjedő kutatásokkal feltölteni, hogy a lehető legjobban leredukálásra kerüljenek a hallucinációk és téves pontozások.

Érdemes továbbá automatizálni valamilyen formában a scrapelést, ami heti szinte összegyűjti a híreket a megadott platformokról, ezeket az ismertetett módon hozzáadja az eddigi adatbázishoz, ellátja weekOfYear paraméterrel, lemmatizálja, leszűri a releváns híreket a kulcsszavak alapján, majd elküldi őket az LLM-nek az analízishez, végezetül a válaszból kinyeri az értékes információkat, amelyek felhasználhatóak a cég számára és az LSTM modell tanítására.

Az LSTM modell előrejelzéseinek és az LLM szentiment analízis rendszerének együttesen a célja egy olyan DTR megalkotása, ami előrejelzések és hírek elemzése mellett komplett alternatív terveket kínál a cégnek, amely kiválthatja a mostani alternatív stratégiákat alkotó jól, de rettentően erőforrás költségesen működő módszerüket. A fejlesztett rendszer jövőbeli célja, hogy a piac szélsőséges változásait is képesek legyenek agilisan követni és tudjanak egy olyan alternatív tervre átállni rövid időn belül majd aztán követni, amely a piac jelenlegi állapotának megfelelő.

Az LSTM előrejelző és LLM hírelemző rendszer kritikus továbbfejlesztései közé tartozik még, hogy a cégnek ne csak a gabonapehely divízióján, hanem az összes többi divíziójára kiterjedően, képes legyen előre jelezni és híreket elemezni az új divíziók eladási adatik és relevanciáik szerint.

ÁBRAJEGYZÉK

1. Ábra – A cég által tervezett stratégia Forrás: Saját szerkesztés	9
2. Ábra - A rendszer "Black box" alapú működése Forrás: Saját szerkesztés	10
3. Ábra - Illusztráció egy mély neurális hálózat architektúráról Forrás: [3] nyomán saját szerkesztés	11
4. Ábra - Illusztráció egy rekurens neurális hálózata architektúráról Forrás: [9] nyomán saját szerkesztés	13
5. Ábra - Az LSTM cella felépítése Forrás: [11] nyomán saját szerkesztés	14
6. Ábra - Kaszkádosított cellák Forrás: [11] nyomán saját szerkesztés	16
7. Ábra - A BiLSTM működése. Forrás: Saját szerkesztés.....	17
8. Ábra - A mondat feldolgozása hagyományos LSTM segítségével. Forrás: Saját szerkesztés	18
9. Ábra - A BiLSTM működése nyelvi feldolgozásnál. Forrás: Saját szerkesztés	18
10. Ábra - Az LLM-ek működésének ábrázolása Forrás: Saját szerkesztés	21
11. Ábra - A RAG rendszer általános működésének bemutatása Forrás: Saját szerkesztés	26
12. Ábra - A prototípus szoftver valós és előrejelzett adatai egymáshoz viszonyítva Forrás: Saját szerkesztés	27
13. Ábra - Auchan eladások Kullback-Leibler [20] eloszlása Forrás: Saját szerkesztés	28
14. Ábra – Spar eladások Kullback-Leibler [20] eloszlása Forrás: Saját szerkesztés.....	29
15. Ábra – Tesco eladások Kullback-Leibler [20] eloszlása Forrás: Saját szerkesztés	29
16. Ábra – Boltok Kullback-Leibler [20] normális eloszlása Forrás: Saját szerkesztés	29
17. Ábra - A prototípus szoftver valós és előrejelzett adatai egymáshoz viszonyítva Forrás: Saját szerkesztés	30
18. Ábra, Az LSTM modell elvárt működése a hírek értékelésének figyelembevételével Forrás: Saját szerkesztés.....	32
19. Ábra - Az origo.hu oldal archív adatbázisának keresője (Forrás: Origo.hu – Saját szerkesztés).....	33
20. Ábra - Az origo.hu oldal szűrőjének manipulálása Forrás: Saját szerkesztés.....	34
21. Ábra - A lementett cikkek linkjeinek összessége (részlet) Forrás: Saját szerkesztés	34
22. Ábra - Az oldal HTML kódja, ami tartalmazza a metaadatokat (részlet) Forrás: Origo.hu - Saját szerkesztés	35

23. Ábra - Felnőtt tartalomként megjelölt cikkek nyitó oldala Forrás: Saját szerkesztés	35
24. Ábra - A strukturált JSON formátum, amibe a metaadatok kerültek Forrás: Saját szerkesztés	36
25. Ábra - A letöltött JSON fájlok összessége a program futását követően (részlet)Forrás: Saját szerkesztés	36
26. Ábra A cikkek csoportosítása, a belőlük készült adatbázis Forrás: Saját szerkesztés	37
27. Ábra Lemmatizálás a gyakorlatban Forrás: Saját szerkesztés.....	38
28. Ábra A gabonapehely szóhoz legközelebb eső 10 szó (magát nem beleértve) Forrás: Saját szerkesztés	39
29. Ábra A logolási folyamata az adatbázis szűrésének Forrás: Saját szerkesztés	40
30. Ábra Példa, ami rámutat a releváns cikkek helyes kezelésére Forrás: Saját szerkesztés ...	44
31. Ábra Példa, ami rámutat az irreleváns cikkek helyes kezelésére Forrás: Saját szerkesztés	44
32. Ábra Példa, ami rámutat a modell eseti hibás pontozására / indoklására Forrás: Saját szerkesztés	45
33. Ábra Példa, ami bemutatja a hallucinálást	45
34. Ábra A RAG rendszer folyamatábrája összevetve a csupán LLM-et használóéval Forrás: Saját szerkesztés.....	47
35. Ábra Példa, ami bemutatja a RAG által javított korábbi példát Forrás: Saját szerkesztés.	47
36. Ábra A hírek százalékos eloszlása kategóriánként.....	49

TÁBLÁZATJEGYZÉK

1. Táblázat - Az OHE bemutatása Forrás: Saját szerkesztés.....	19
2. Táblázat - A vektor embedding bemutatása Forrás: Saját szerkesztés.....	20
3. Táblázat, A talált releváns cikkek száma, az alkalmazott kulcsszavak számával összevetve Forrás: Saját szerkesztés.....	40
4. Táblázat A hírek eloszlása kategóriánként. Forrás: Saját szerkesztés.....	49

IRODALOMJEGYZÉK

- [1] J. Schmidhuber, “Annotated History of Modern AI and Deep Learning,” Dec. 29, 2022, *arXiv*: arXiv:2212.11279. doi: 10.48550/arXiv.2212.11279.
- [2] J. Heaton, “Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning,” *Genet. Program. Evolvable Mach.*, vol. 19, no. 1, pp. 305–307, Jun. 2018, doi: 10.1007/s10710-017-9314-z.
- [3] Z. Lipton, “A Critical Review of Recurrent Neural Networks for Sequence Learning,” May 2015.
- [4] H. Jaeger, “Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach,” *GMD-Forschungszentrum Informationstechnik* 2002, vol. 5, Jan. 2002.
- [5] S. Hochreiter, “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions,” *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 6, pp. 107–116, Apr. 1998, doi: 10.1142/S0218488598000094.
- [6] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990, doi: 10.1109/5.58337.
- [7] C. Tallic and Y. Ollivier, “Unbiasing Truncated Backpropagation Through Time,” May 23, 2017, *arXiv*: arXiv:1705.08209. doi: 10.48550/arXiv.1705.08209.
- [8] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
- [9] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997, doi: 10.1109/78.650093.
- [10] M. Yang *et al.*, “Design and Implementation of an Explainable Bidirectional LSTM Model Based on Transition System Approach for Cooperative AI-Workers,” *Appl. Sci.*, vol. 12, no. 13, Art. no. 13, Jan. 2022, doi: 10.3390/app12136390.
- [11] Y. Bengio, R. Ducharme, and P. Vincent, “A Neural Probabilistic Language Model,” in *Advances in Neural Information Processing Systems*, MIT Press, 2000. Accessed: Apr. 01, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2000/hash/728f206c2a01bf572b5940d7d9a8fa4c-Abstract.html
- [12] “Speech and Language Processing.” Accessed: Apr. 01, 2024. [Online]. Available: <https://web.stanford.edu/~jurafsky/slp3/>
- [13] A. Vaswani *et al.*, “Attention Is All You Need,” Aug. 01, 2023, *arXiv*: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.
- [14] “Llama 3.1 - 405B, 70B & 8B with multilinguality and long context.” Accessed: Nov. 04, 2024. [Online]. Available: <https://huggingface.co/blog/llama31>
- [15] A. Dubey *et al.*, “The Llama 3 Herd of Models,” Aug. 15, 2024, *arXiv*: arXiv:2407.21783. doi: 10.48550/arXiv.2407.21783.
- [16] “Stemming and lemmatization.” Accessed: Nov. 04, 2024. [Online]. Available: <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>
- [17] B. Liu, *Sentiment Analysis and Opinion Mining*. Springer Nature, 2022.
- [18] M. A. Resources Information, *Research Anthology on Implementing Sentiment Analysis Across Multiple Disciplines*. IGI Global, 2022.
- [19] P. Lewis *et al.*, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” Apr. 12, 2021, *arXiv*: arXiv:2005.11401. doi: 10.48550/arXiv.2005.11401.

- [20] J. M. Joyce, “Kullback-Leibler Divergence,” in *International Encyclopedia of Statistical Science*, M. Lovric, Ed., Berlin, Heidelberg: Springer, 2011, pp. 720–722. doi: 10.1007/978-3-642-04898-2_327.
- [21] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, “Stanza: A Python Natural Language Processing Toolkit for Many Human Languages,” Apr. 23, 2020, *arXiv*: arXiv:2003.07082. doi: 10.48550/arXiv.2003.07082.
- [22] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching Word Vectors with Subword Information,” Jun. 19, 2017, *arXiv*: arXiv:1607.04606. doi: 10.48550/arXiv.1607.04606.
- [23] J.-D. Guenette, *Price Controls: Good Intentions, Bad Outcomes*. World Bank, Washington, DC, 2020. doi: 10.1596/1813-9450-9212.
- [24] “OpenAI Platform.” Accessed: Nov. 04, 2024. [Online]. Available: <https://platform.openai.com>