

TDK dolgozat

Czetin Márk

Kovács Hunor György

2024

Komplex piaci adatokon alapuló gépi tanulás alapú agilis forecast döntéstámogató rendszer

Czetin Márk

Kovács Hunor György

Széchenyi István Egyetem

Gépészmérnöki, Informatikai és Villamosmérnöki Kar

Mérnökinformatikus szak, BSc 3. év

Témavezető:

Hajdu Csaba, egyetemi tanársegéd

dr. Bognár Tamás, egyetemi adjunktus

Kézirat lezárásának dátuma: 2024.04.02

Absztrakt

Komplex piaci adatokon alapuló gépi tanulás alapú agilis forecast döntéstámogató rendszer

Az elmúlt évek globális eseményei jelentősen megváltoztatták a fogyasztói szokásokat. Kifejezetten a nagyobb vállalatok csak jelentős erőforrások igénybevételével tudják előrejelezni az eladásait.

További nehézséget okoz, hogy az adatáramlás egy adott vállalat és az értékesítő csatornák között sokszor hiányos, akár nem is létező. Következésképpen még nehezebbé válik a jövőbeli eladások megbecslése a sell out (külső értékesítési) adatok hiányában. A kutatásban együttműködő nagyvállalat is ezeket a nehézségeket fedezte fel.

Jelenlegi kutatás célja, egy gépi tanulás alapú rendszer kiépítése, amely megfelelően képes előrejelezni az eladásokat különböző bemeneti paraméterek alapján (pl. az év aktuális hete, promóció megléte egy terméken, a termék célközönsége). Elvárás, hogy legalább olyan, vagy pontosabb előrejelzéseket adjon, mint amilyeneket jelenleg a vállalatnál dolgozó szakemberek szolgáltatnak jelentős idő és pénzületi erőforrások felhasználásával.

A modell elkészüléséhez megfelelő minőségű és mennyiségű adat szükséges. A titkosított adatokat a kutatáshoz a nagyvállalat biztosítja. Ezek az adatok nagyon sok helyen hiányosak, nem egységesek bizonyos értékesítési csatornákon, így, a dolgozat egy jelentős része foglalkozik az adatok feldolgozásával.

A gépi tanulás alapú modell elkészítéséhez egy Long Short-Term Memory (LSTM) architektúrára épülő modell került kiválasztásra. Ez lehetőséget ad az adathalmaz szekvenciális feldolgozására, amely jelentősen növelheti a modell pontosságát amikor historikus adatok alapján történik a jövőbeli előrejelzés.

A kutatás a megvalósíthatóság és adatfeldolgozás területén már jelentős eredményekkel rendelkezik, amik számos jövőbeli fejlesztésnek nyújtanak alapot. Jelenleg a keretrendszer képes előre jelezni az eladásokat, a cél, hogy a jövőben komplett alternatív stratégiákat előállító döntéstámogató szolgáltatás készüljön el. Az adatok a nagyvállalat egy divíziójából származnak, azonban jövőbeli fázisban cél, hogy minden divízió számára elérhető legyen a keretrendszer.

Köszönetnyilvánítás

Hálával tartozunk felkészítő tanárainknak, Hajdu Csaba és dr. Bognár Tamás Tanár Úrnak áldozatos munkájukért, amit a dolgozat elkészülése során tanúsítottak. Észrevételeik, motiválásuk és szakmai tapasztalatuk nagyon sokat hozzatett 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	13
3.2.2	Az LSTM modell neuronjai	14
3.2.3	Az LSTM modell kapujai.....	15
3.2.4	Bidirectional LSTM	17
3.3	Természetes nyelvi feldolgozás.....	20
3.3.1	Szavak értelmezése gépi tanuláson keresztül.....	20
3.3.2	Nagy nyelvi modellek	22
3.3.3	RAG rendszer	24
4.	Prototípus szoftver.....	27
5.	Az előrejelzéshez fejlesztett szoftver	29
5.1	Az eladási adathalmaz	29
5.2	A szoftver felé támasztott igények	32
5.3	Szoftver architektúra	33
5.4	Valós idejű hírek feldolgozása RAG-on keresztül.....	36
6.	Az eredmények.....	38
6.1	A tanítás folyamata.....	38
6.2	Az előrejelzés pontossága	44

7. Konklúzió.....	47
7.1 Jövőbeli munka	47
ÁBRAJEGYZÉK	50
TÁBLÁZATJEGYZÉK.....	52
IRODALOMJEGYZÉK.....	53

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, logisztikát, 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 befolyásoltság 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, agilisan 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 agilisan 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 tanulási 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 termékcsoporth Lengyelország forgalmának körülbelül a 40%-át teszi ki, míg ugyanez a termékcsoporth 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 (Magyarországi Gabonapehely Divízió) igényeire specializálva alkalmazni ezeket, muszáj összehangolva használni őket, ami a divíziók külön-külön teljesítésére negatív hatással lehet.

2.2 Agilis működés a vállalatban belül

A GD egy Fast-Moving Consumer Goods (FMCG) termékkategória, 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 előnyé kovácsolható 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, ehelyett könnyen adódik a tervek nem teljesüléséért a végrehajtás tervezés „hibáját” keresni. Tovább nehezíti ezt a feladatot a folyamatos együttműködés igénye a vállalatban és a klaszterben belül is.

Ez különösen veszélyes a GD-re nézve, mivel főként 2 termékcsoportja 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 termékkategóriából áll, így, ha az egyik termékcsoport 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 vállalatban 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

csomagolóanyagot 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.

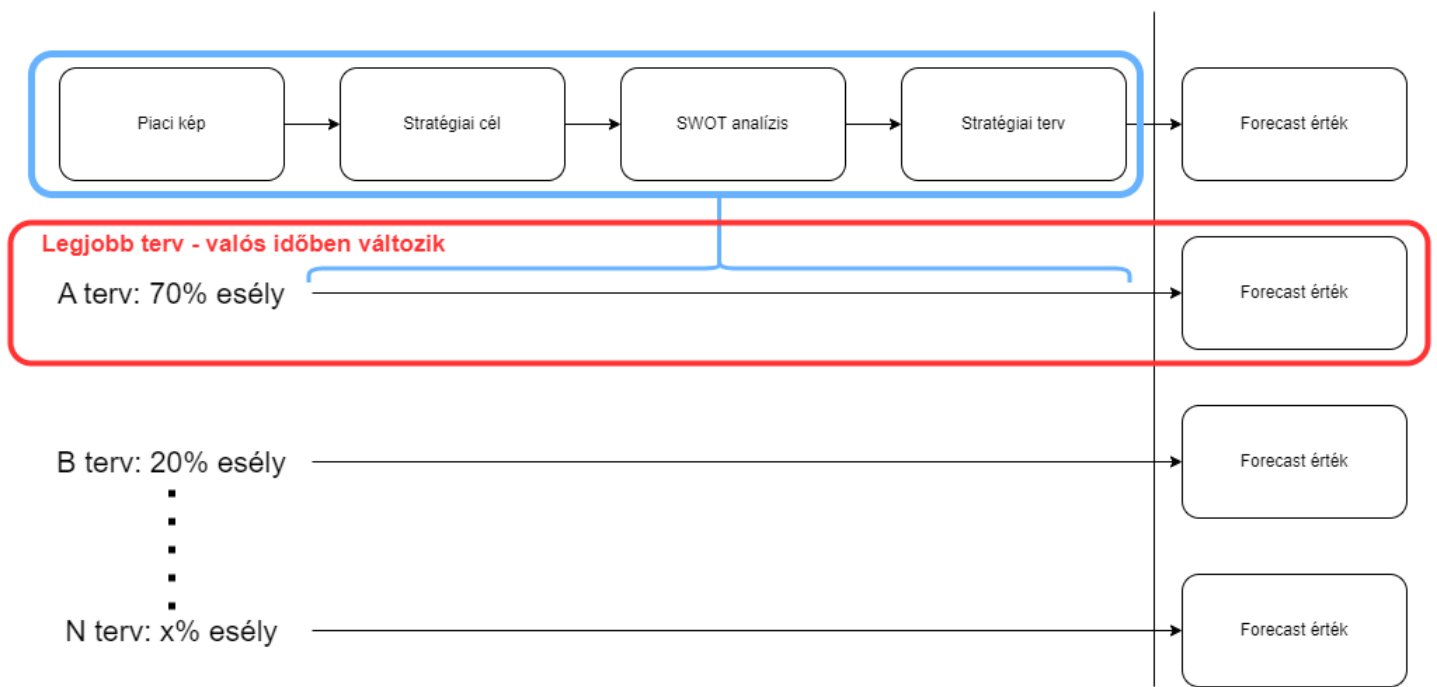
Megkülönböztethető éves és havi forecastolás. Az éves forecastolás egy hónapokon át tartó tervezési folyamat, aminek célja, hogy a GD az egész évre megbecsülje előre a várható eladásokat. Ezt megteszi a klaszteren belül lévő többi ország képviselője is, és feltöltik az informatikai rendszerbe, amin keresztül 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 májustól körülbelül szeptemberig tart, ahol az egész éves eladásukat előre megbecsülik. 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 kiszerelésének 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 kiszerezésre 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 becsülnek 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 folyamatos 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.



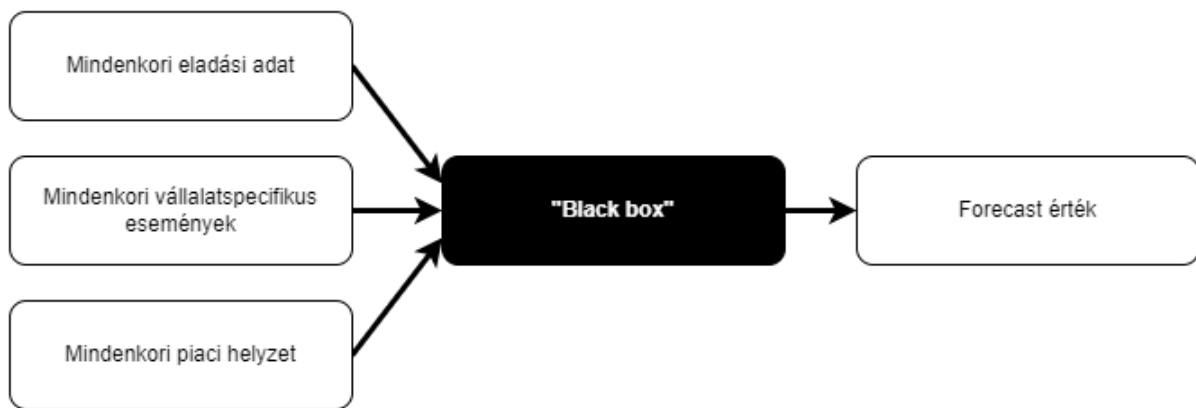
1. Ábra – A cég által tervezett stratégia Forrás: Saját szerkesztés

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

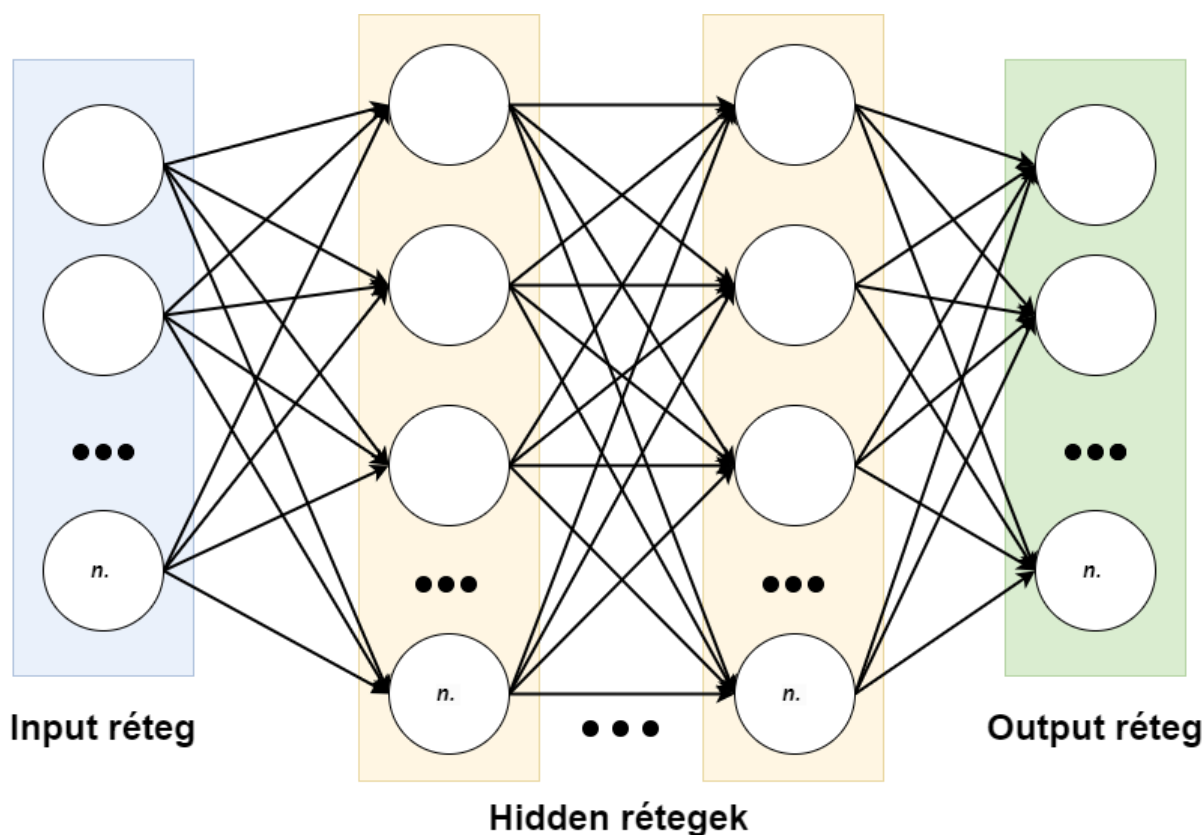
Ahhoz, hogy egy ilyen modell ismerhesse a piaci trendeket és megfelelően tudjon működni, gépi tanulásra van szükség. Ez azt jelenti, hogy a szoftvernek kialakul egy belső értékelési rendszere, ami képes úgy feldolgozni a bemeneti paramétereket, hogy a kimeneti közelítése az elvárthoz közel legyen.

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 kiszűrésre 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 visszaterjesztést 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 feldolgoznia 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.

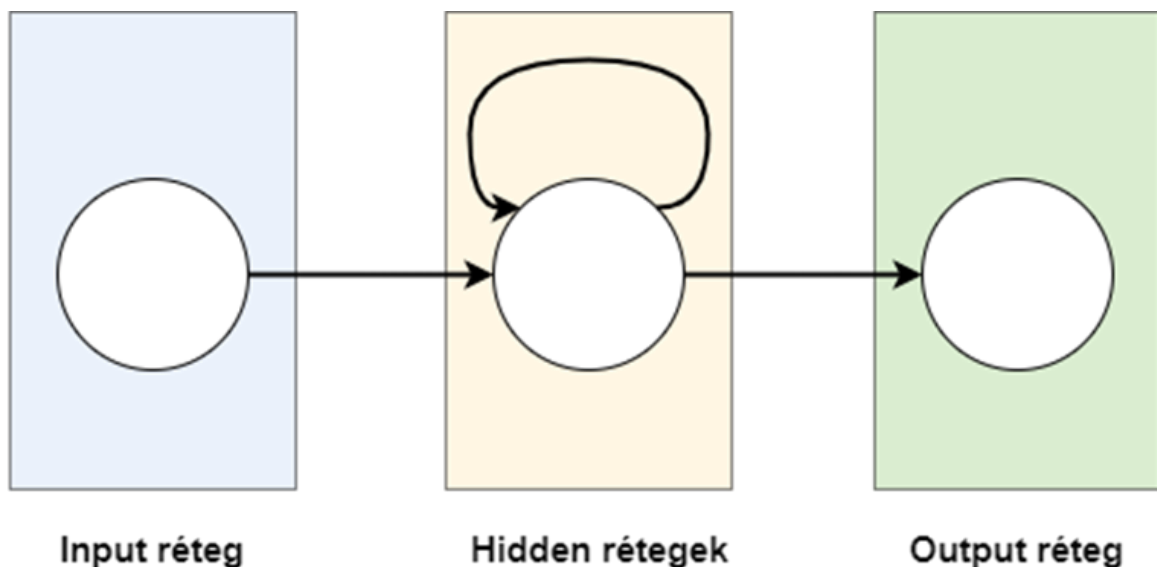
3.2 A választott architektúra ismertetése

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 tanulási modellek esetén ez a működés úgy érhető el, ha egy rekurrens modellt választunk (4. Ábra)[3], [4]. Elsődlegesen ez abban különbözik egy hagyományos neurális háló felépítésétől, hogy a neuronok rendelkeznek még egy önmagukba visszacsatoló éllel, amelynek szintén 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] 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.

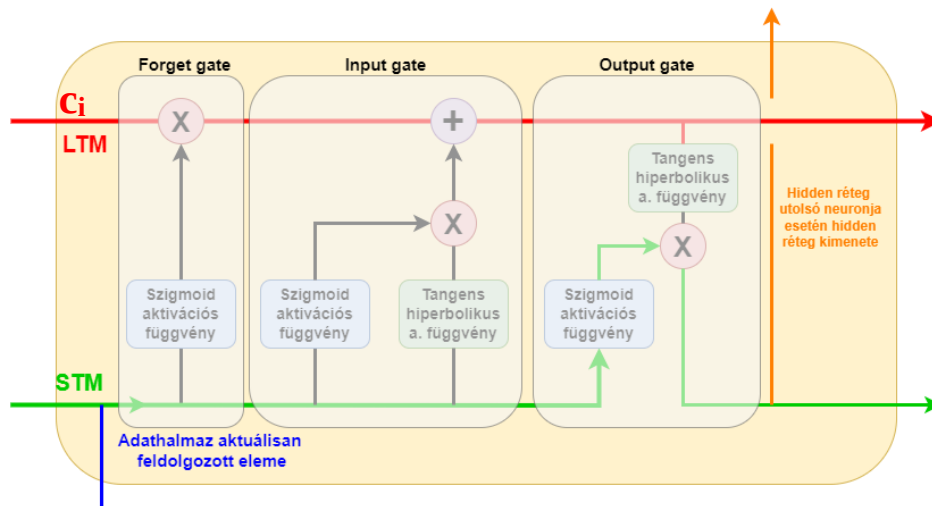
3.2.2 Az LSTM modell neuronjai

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
- Input gate
- Output gate



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, mint ahogy a képlet is mutatja [8] alapján.

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

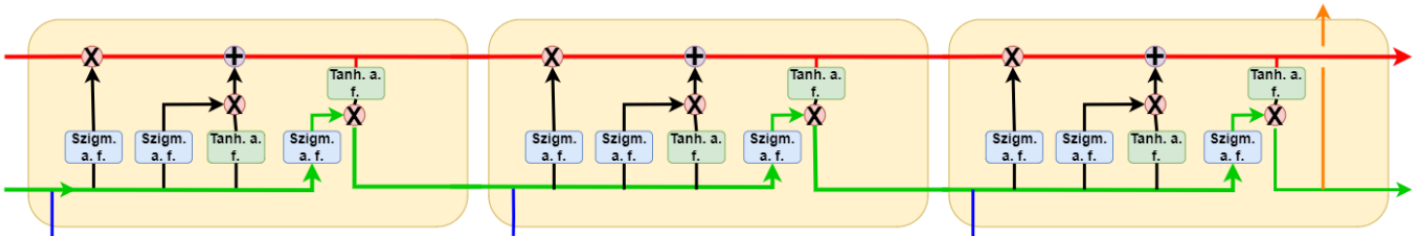
Az input gate fogadja az adathalmaz aktuális bemenő input értékét és az STM szálon érkező értéket. Ezeket felhasználva az input gate kiszámítja, hogy milyen potenciális értéket adjunk hozzá az LTM-hez, és ennek a potenciális értéknek hány százalékát adjuk hozzá. Ehhez a kapu 2 aktivációs függvénnyel 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 megszorzódik 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 [8] alapján, ami egyúttal az új LTM érték, amit a következő cella forget gate-je és az aktuális output gate fog megkapni, mint bemenet.

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \quad (2)$$

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 megszorzódik a súlyozásukkal, majd összeadásra kerülnek, és a végén hozzáadódik az él eltolása. A második aktivációs függvény a tangens hiperbolikus, aminek bemenete az LTM 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 [8] alapján.

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).

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \quad (3)$$

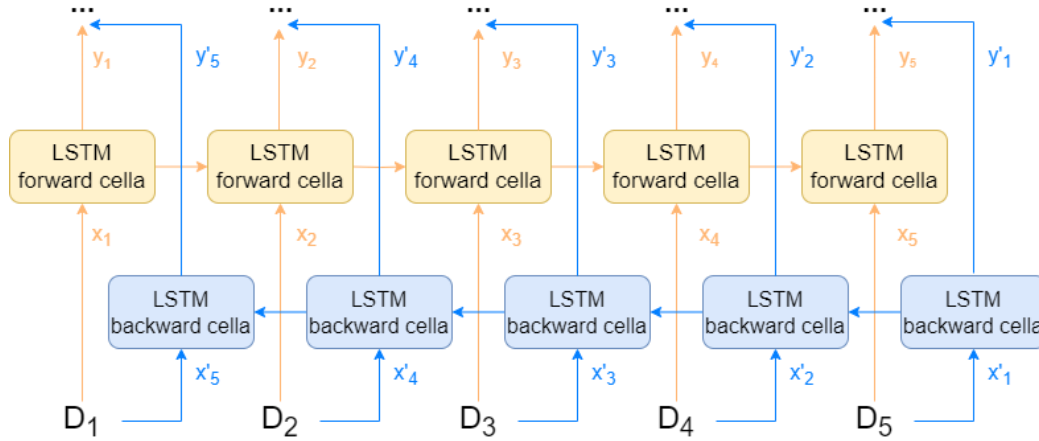


6. Ábra - Kaszkádosított cellák Forrás: [11] nyomán saját szerkesztés

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 „tekintsen a jövőbe”, hogy megtudjon határozni köztes értékeket. A kutatás alatt ez a sell out 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 megakarjuk 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)



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 [2] alapján, így ez:

$$F_1 = [f_1, f_2] \quad (4)$$

$$B_n = [b_{n-1}, b_n] \quad (5)$$

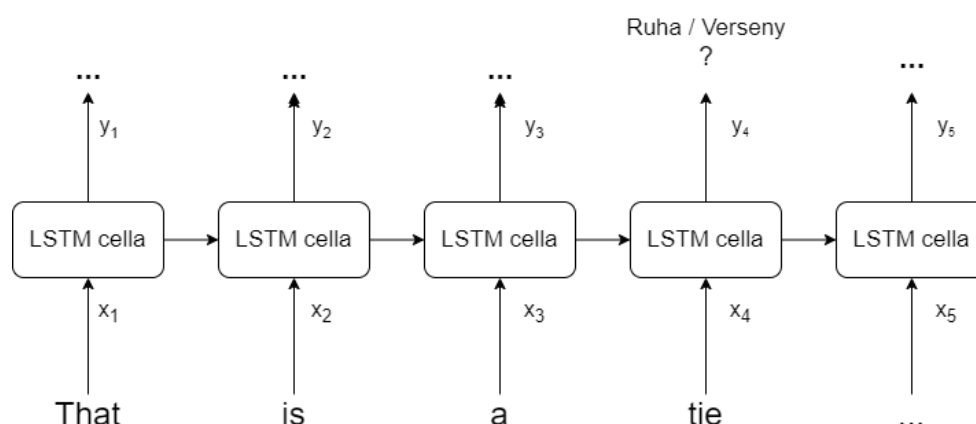
$$K_1 = [f_1, f_2, b_{n-1}, b_n] \quad (6)$$

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)

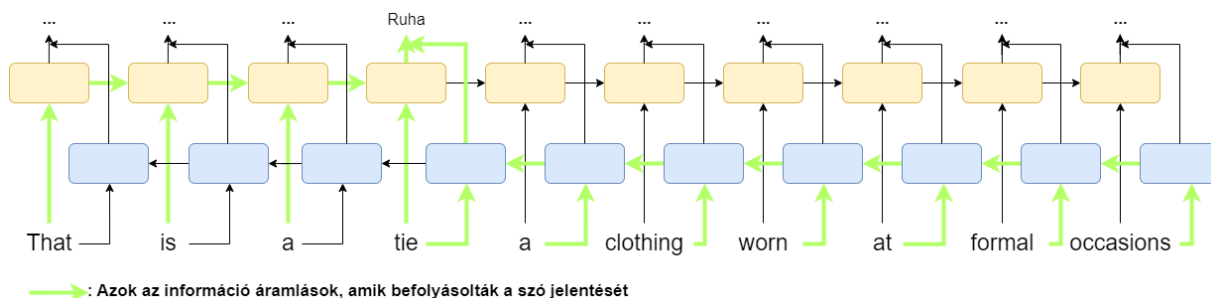
That is a tie, the tournament can't be decided yet.
(Ez döntetlen, így nem lehet befejezni a tornát)

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 rendelkezhesen 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 lásd: 20. oldal)



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 vector embeddingek 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, hosszúkas. 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 Lásd: 22. oldal) 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
Égettest	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
Hosszúak	0	0.8	0

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

A szavak közötti műveletek is alátámasztják a vector 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.

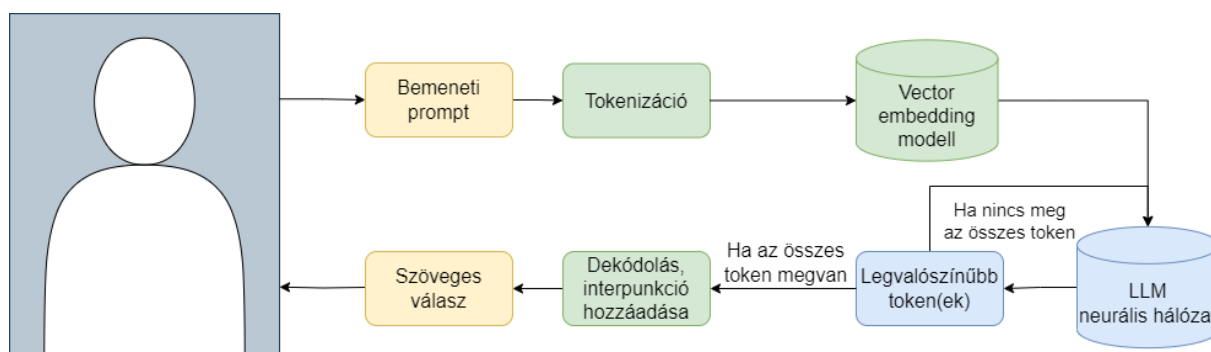
A fentebb leírt működések természetesen akkor valósulnak meg, ha megfelelő módon kerültek tanításra a vector embedding modellek. Ezeknek a tanítása hatalmas mennyiségű folyószövegen történik, ahol a szavak közötti logikai kialakulás elsődlegesen a szavak kontextusától függ. Minél többször fordul elő egy szó egy másik közelében, annál inkább logikailag összekapcsoltak 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 tárolásához, továbbá úgyszint nagyon nagy mennyiségű számítási kapacitás szükséges. Ehhez jellemzően 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 még.

A média által lehozott cikkek 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 akadálnak a leküzdésére egy LLM megfelelő lehet, hiszen párhuzamot képes vonni egy mondat értelme, és annak a gazdasági behatása között, amit megfelelő utasítások hatására képes számszerűen értékelni.

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 vector embedding modell segítségével, 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 logikusan 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 meghatározása. Miután a szöveg készen áll, ez visszajuttatásra kerül a felhasználóhoz (10. Ábra).

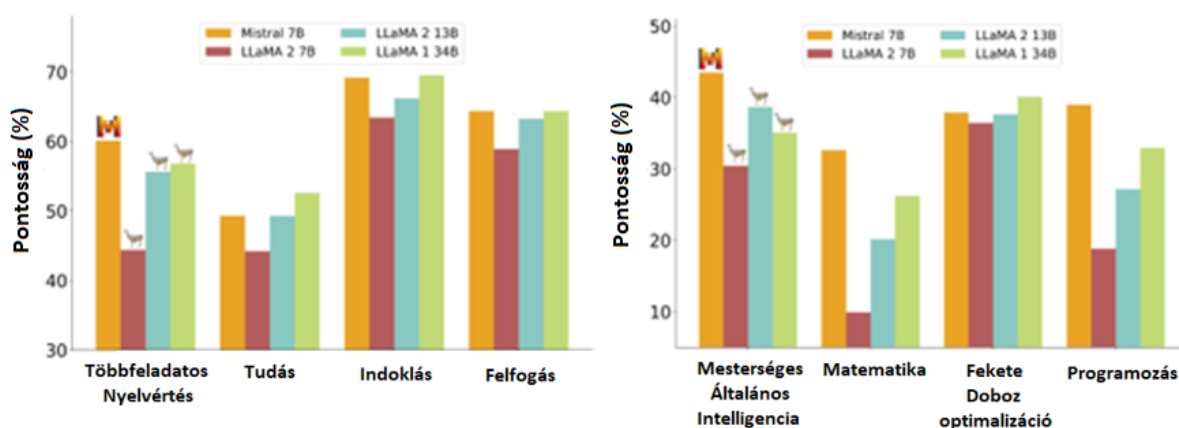


10. Ábra - Az LLM-ek működésének ábrázolása Forrás: Saját szerkesztés

Az LLM-ek jelenleg a felhasználók számára két formában érhetőek el: online szervízeket vesznek igénybe, mint például az OpenAI által szolgáltatott chatGPT, ahol a felhasználók kérése egy API híváson keresztül jut el az OpenAI szervereihez, ahol az ott futó LLM erre választ ad, majd visszajuttatja a felhasználóhoz az oldalon keresztül. Ennek előnye, hogy a számítási kapacitást a cég szolgáltatja, amiért fizetni kell, azonban így ezzel nem kell a továbbiakban foglalkozni. Hátránya, hogy az adatok kikerülnek saját kézből, és a kutatás jellegéből adódóan, ezek érzékeny adatok, illetve ezenkívül nincs lehetőség az LLM működését semmilyen módon befolyásolni a kutatás folyamán, mivel az privát. A másik megközelítés, a különböző open-source LLM modellek lokális futtatása. Ebből számos elérhető változat létezik, különböző paraméter számmal, teljesítménnyel stb. Ezeknek előnye, hogy saját kezűleg lehet őket tovább tanítani egy adott témán, a kutatás esetén gazdasági híreken, illetve gazdasági szakirodalmakon. További előny, hogy az adatok végig saját kézben maradnak, nem kerülnek

ki harmadik félhez feldolgozásra, ami adatbiztonság és a projekt szempontjából rendkívül jelentős. Hátrányuk, hogy a futtatásuk saját erőforrást igényel, mivel ezt a felhasználónak kell szolgáltatnia saját magának.

A kutatás jelenleg abba az irányba tart, hogy egy jól teljesítő open-source modell kerüljön további tanításra és felhasználásra, mivel ennek az iránynak a pozitívumai bőven ellensúlyozzák a negatívumait. A modell kiválasztása jelenleg még nem történt meg, azonban előreláthatólag a Mistral7B, vagy a Mistral modell család egyike lesz a kiválasztott LLM. (11. Ábra)[14]



11. Ábra - A Mistral7B teljesítménye diagramon ábrázolva egyéb state of the art modellekkel szemben. Forrás [16]

3.3.3 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 tanulását). Ez amiatt van,

mert a kis paraméter szám mellett limitált súlyozásokkal és eltolásokkal rendelkeznek, 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. [15]

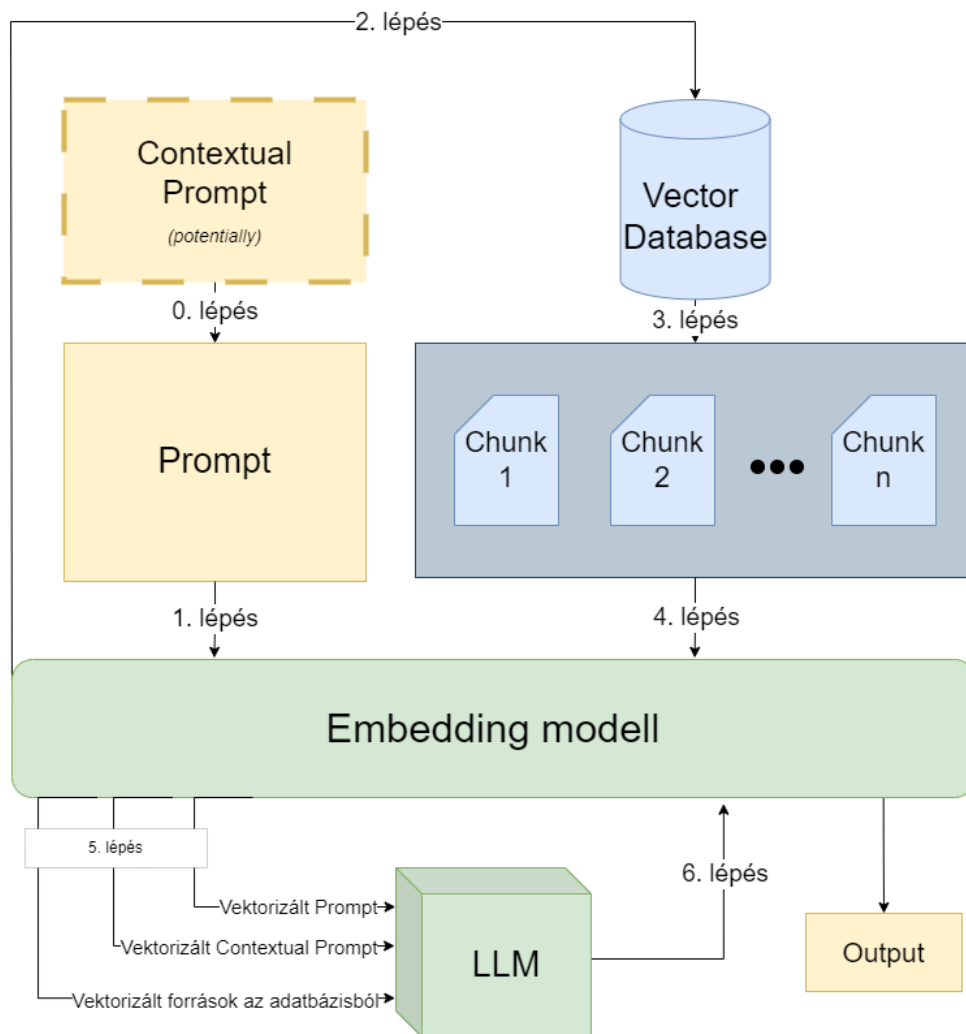
További előnye még a RAG rendszereknek, hogy kiegészítik az LLM-ek generális tudását vállalat specifikus tudással. Tehát, ha a cég rendelkezik valamelyik különleges sajátossággal, ami bizonyos módon befolyásolja a hírek értékelését, akkor az LLM már nem csak a saját tudásával (ami a súlyokból származik), 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 vector 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 vector 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, aminek a kiszámítása a koszinusz tétel alkalmazásával lehetséges.

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|} \quad (7)$$

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. (12. Ábra)

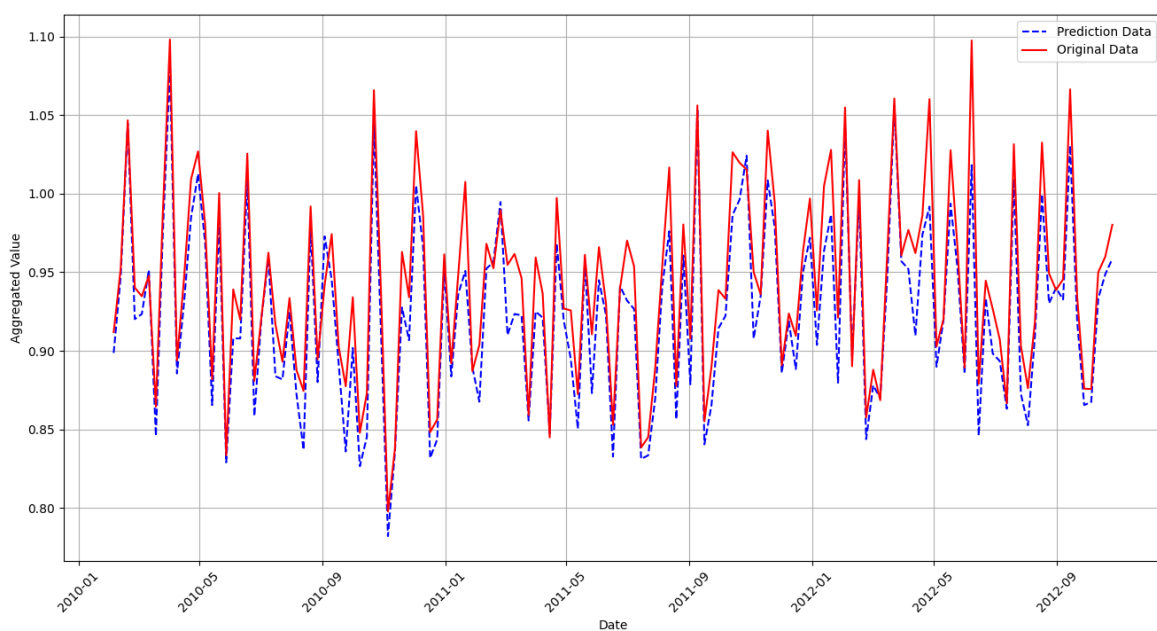
Jellemzően a prompt kiegészítésre kerülhet úgynevezett „contextual prompt” -al, aminek a célja az LLM viselkedésének további irányítása a fejlesztők által, amivel plusz információval látják el az LLM-et a felhasználóról, a feladatáról, az elvárt formátumú válaszról stb.



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

4. Prototípus szoftver

2023 őszi félévében a tudományos diákköri konferencia (TDK) alkalmával elkészült dolgozat a mostani értekezés prototípusának szolgált. Az ezt megelőző dolgozat célja, hasonló volt mint a mostaninak, azonban ott még nem a valós vállalati adatokkal dolgoztunk, hanem egy előre megtisztított, rendszerezett és a szerző által már felülvizsgált hasonló adatbázissal. A tavalyi dolgozat bemutatta azt, hogy az LSTM architektúra alapú neurális hálók alkalmasak arra, hogy előrejelezzünk eladási adatokat (13. Ábra). Az ottani eredmények alapján vittük tovább ebbe a dolgozatba az LSTM architektúrát. A jelenlegi dolgozat jól körüljárja az adatfeldolgozás problémáját is, mivel ezzel a kihívással tavaly nem kellett foglalkoznunk, azonban a valós nyers adatok értelmezése, tisztítása és feldolgozása során világossá vált, hogy ez a tényező nagyban befolyásolja a feltanítás és előrejelzés minőségét is.



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

A jelenlegi prototípus a korábbi eredmények és módszerek alapján épült fel, azokat továbbfejlesztve és finomítva. A jelenlegi szoftver funkcionalitása kiterjed az előző verzió által nyújtott összes funkcióra, és képes ugyanazokat a mérőszámokat alkalmazni, mint az előző változat. Ennek köszönhetően a fejlesztés során előzetesen tesztelt és validált metrikákat használhatunk, amelyekkel összehasonlíthatjuk az új eredményeket. Emellett a korábbi

tapasztalatok és eredmények felhasználása lehetővé teszi a hatékonyabb fejlesztést és a korábbi hibák elkerülését.

5. Az előrejelzéshez fejlesztett szoftver

5.1 Az eladási adathalmaz

A velünk együttműködő vállalat az elmúlt 1 év eladási adatait szolgáltatja a szoftver elkészüléséhez. Az elsődleges kihívást az jelenti, hogy vannak olyan boltok (vizsonteladók), akik miután vásárolnak gabonapehely termékeket a vállalattól (sell in vagy belső eladás), nem szolgáltatnak adatokat arról, hogy ők ebből mennyit adnak el a vásárlók felé az üzleteikben (sell out vagy külső eladás). A vállalat 20 vizsonteladó üzletnek értékesít, és ebből 3 olyan üzlet van (Tesco, Spar, Auchan), amelyek szolgáltatnak visszafele is sell out adatot a vállalat számára. A belső és külső eladások kilóban értendők. A tanítás alapötlete is ehhez kapcsolódik, mivel megvizsgáltuk, hogy ezeknek a szolgáltatott sell out adatoknak milyen eloszlásuk van, és abból indultunk ki, hogy ha az eloszlás megegyezik, akkor feltételezhetően a többi vizsonteladó eladásának az eloszlása is hasonló tendenciát fog mutatni. Ha az előbb említett hipotézis beigazolódik, akkor a feltanított sell in – sell out adatok alapján képesek leszünk olyan vizsonteladók eladásait is előre jelezni, akik nem nyújtanak információt arról, hogy mennyi terméket adnak el a vásárlók felé.

Az adatok feldolgozása és átalakítása időigényes feladatnak bizonyult. A vállalat a Systems Applications and Products (SAP) rendszerükből szolgáltatja nekünk a sell in (3. Táblázat Lásd: 30. oldal) eladásokat termékenként, heti bontásban. Ezeket az adatokat a Python [16] programozási nyelv és azon belül a pandas [17] valamint numpy [18] könyvtár segítségével dolgoztuk fel. Termékenként sorba rendeztük hetekre lebontva az adatokat és mellé rendeltük, hogy melyik vizsonteladó mennyit vásárolt a vállalattól, ezen kívül elláttuk a megfelelő paramétereivel a terméknek (termék neve, év, hét, ár, kiserelés mennyisége, volt-e az adott időszakban online vagy televíziós marketing).

Product Name	Year	Week Of The Year	Packaging Size g	List Price	TV Marketing	Online Marketing	Shop 1_Sell In	Shop 2_Sell In	Shop N_Sell In
Product 1	2023	20	425	1100	1	0	52.84	92.2	23.4
Product 1	2023	21	425	1100	0	0	28.2	61.8	16.82
Product 2	2023	20	375	975	1	1	130.9	126.7	94.5

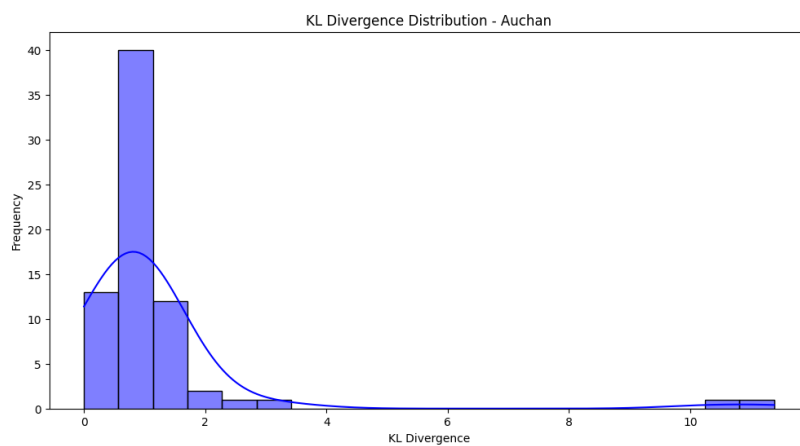
3. Táblázat - A sell in adatok (részlet az adathalmazból) Forrás: Saját szerkesztés

Ezután feldolgoztuk a kapott sell out (4. Táblázat) adatokat is, kiegészítve a sell in adatokkal. Ebben az esetben a Spar és a Tesco eladásai már eleve heti bontásban voltak, az Auchan adatait viszont átlagolni kellett, mivel ők havi bontásban szolgáltatnak adatot (ez a lentebb felsorolt ábránál is látszik). 72 termékünk van, 53 hét tartozik mindegyikhez, így a végeredmény egy 3816 soros és 49 oszlopos adathalmaz lett.

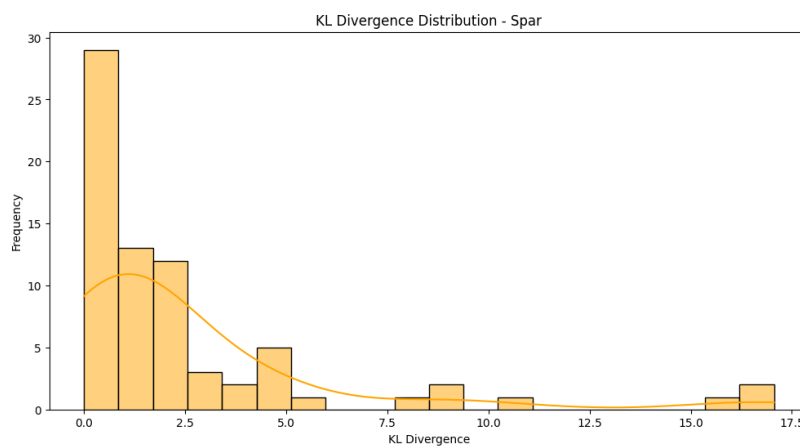
Product Name	Year	Week Of The Year	Packaging Size g	List Price	TV Marketing	Online Marketing	Shop 1_Sell out	Shop 2_Sell out	Shop N_Sell out
Product 1	2023	20	425	1100	1	0	45.23	84.2	93.4
Product 1	2023	21	425	1100	0	0	38.2	12.8	64.82
Product 2	2023	20	375	975	1	1	96.9	31.7	81.5

4. Táblázat - A sell out adatok (részlet az adathalmazból) Forrás: Saját szerkesztés

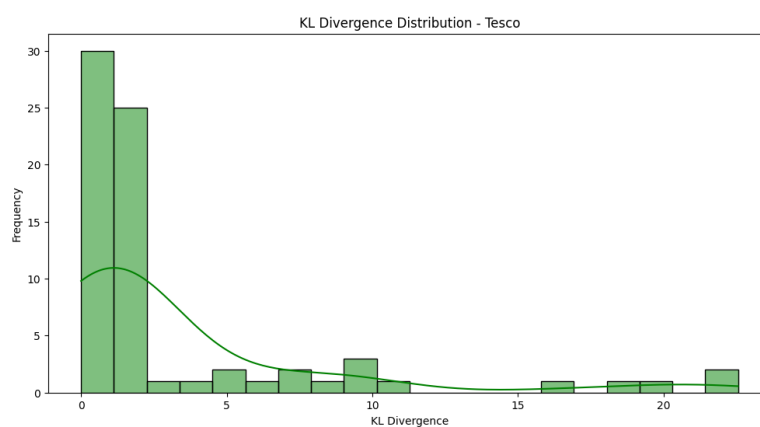
Elsődlegesen megvizsgáltuk a sell in és sell out adatok arányát minden termékre a fentebb említett Auchan (14. Ábra Lásd: 31. oldal), Spar (15. Ábra Lásd: 31. oldal), Tesco (16. Ábra Lásd: 31. oldal) boltokban Kullback–Leibler divergencia (KL Divergencia) [19] segítségével, hogy megbizonyosodjunk arról, hogy a sell out adatok generálása lehetséges-e.



14. Ábra - Auchan eladások Kullback-Leibler [19] eloszlása Forrás: Saját szerkesztés

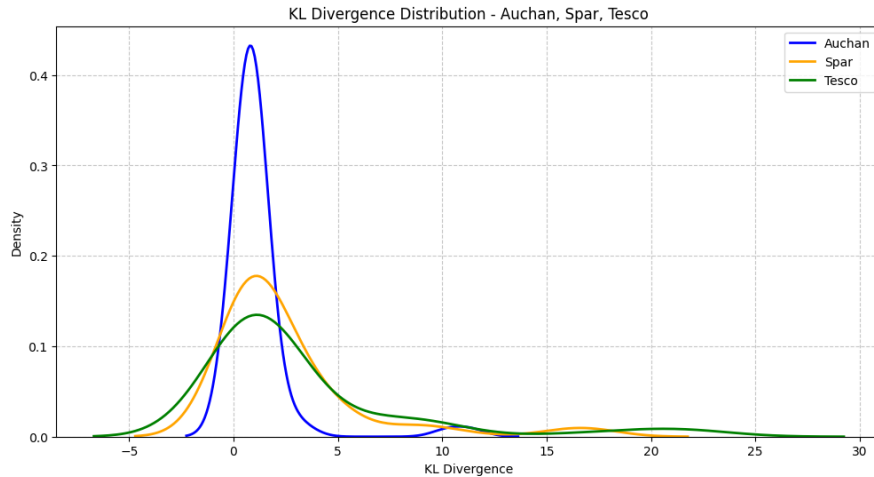


15. Ábra – Spar eladások Kullback-Leibler [19] eloszlása Forrás: Saját szerkesztés



16. Ábra– Tesco eladások Kullback-Leibler [19] eloszlása Forrás: Saját szerkesztés

Ezután vizsgáltuk a kapott KL számok eloszlását és megállapítottuk, hogy az eloszlás nagy mértékben hasonló (17. Ábra) és a sell out adatok generálása feltételezhetően elvégezhető a többi boltra levetítve is.



17. Ábra– Boltok Kullback-Leibler [19] normális eloszlása Forrás: Saját szerkesztés

5.2 A szoftver felé támasztott igények

Követelmény, hogy a szoftver be tudjon fogadni egy adathalmazt a tanulás, validálás, és előrejelzés végett. Kell, hogy legyen megfelelő mennyiségű adatunk, és karakterisztikánk, mivel a szoftver csak így tud megfelelő mintákat keresni, hogy később pontos előrejelzéseket hajtson végre. Az adatokat át kell tudnunk alakítani a megfelelő formátumra, ez esetünkben $[0;1[$ közé eső skálázást, és one-hot encoding alapú kódolást jelent, az eladási adatok mennyiségét, illetve helyét illetően. Az átalakított adatokat tovább kell tudnunk LSTM formátumra alakítani ezután két egységre bontjuk az adathalmazt, egy tanítási és egy tesztelési blokkra. Ezt követően a két halmazunk alakját LSTM formátumra kell módosítani, ez tartalmazza az adat mennyiségét, és mennyi karakterisztikát használva tanítjuk fel a modellünket. A szoftvernek tudnia kell paramétereizhetően (batch méret, epoch szám, hidden unit number, learning rate) feltaníthatónak lennie, tehát a paraméterek módosításával változnia kell a fentebb felsorolt adatoknak is, és vele együtt a tanulás minőségének is. A tanítás közben el kell tudnunk tárolni az adatokat a tanítási és a validációs görbéről. A tanulás végeztével elmentjük a modellt, és grafikonon kirajzoljuk a tanulási és validációs görbe egymáshoz való viszonyát és tendenciáját a használt epoch számok függvényében. A grafikonok lementésre kerülnek, mivel az egymáshoz való relatív viszonyításuk tükrözi a modell tanításának

minőségét. A feltanított modellel elvégezzük az előrejelzéseket. Fontos lépés, hogy visszaalakítjuk az adatokat az eredeti skálájuk szerint a $]0;1[$ -es tartományból, majd ezek az adatok lementésre kerülnek egy külön fájlba. Ezután a valós adatokból és az előre jelzettek közül Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R^2 értékeket számolunk [20] alapján, és az így kapott eredmények lementésre kerülnek egy külön fájlba. A képletekben az $\hat{y}$ az előrejelzett értéke az y -nak.

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2 \quad (8)$$

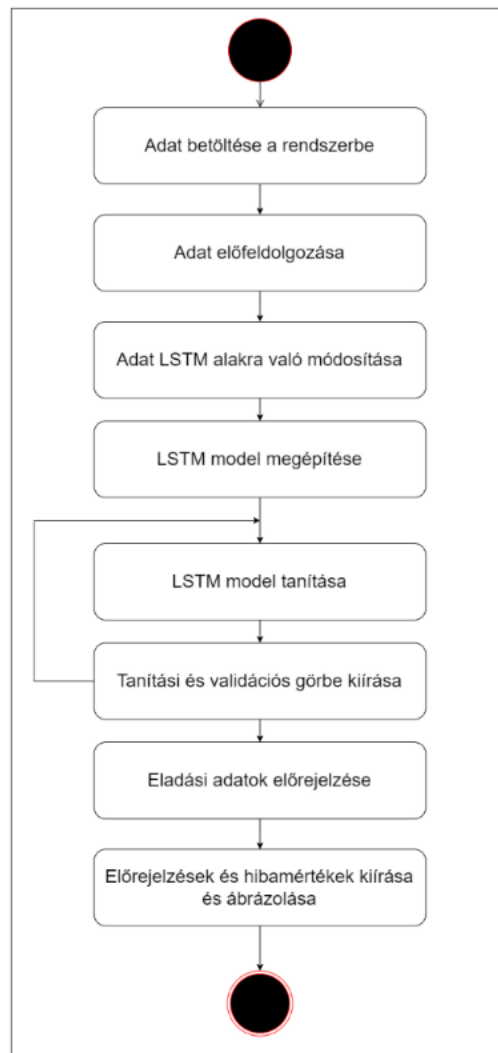
$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2} \quad (9)$$

$$R^2 = 1 - \frac{\sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2} \quad (10)$$

Kiíratjuk grafikonon az egész adathalmazt (előrejelzett és valós adatok) az eladás szerint a sorba rendezett dátumok függvényében, amit ezután kép formájában lementünk. Összehasonlítás végett aggregáljuk a valós és előrejelzett adatokat, hogy egymáshoz lehessen hasonlítani őket, majd a kapott eredményt a hetek függvényében kirajzoljuk egy grafikonon és lementjük kép formájában. A szoftvernek az előzetes X_test adatokon kívül, tudnia kell más módon forrásból származó (de ugyan olyan formára hozott) adatokon is helyes előrejelzést adnia (különböző bolt sell in adatát a megfelelő termék paramétereirez rendelve is helyes előrejelzést kell adnia a sell out adatról).

5.3 Szoftver architektúra

Az LSTM alapú előrejelzésért felelős rendszer a Visual Studio Code integrated development environment (IDE) használatával készült Python nyelven. Az IDE-n belül Jupyter Notebook környezetben került fejlesztésre a szoftver, mivel támogatja a cella alapú futtatást, szövegkiemelést, fájl előnézet-megnyitást, kód kiegészítést és egyéb kényelmi funkciókat. A szoftver alapkövetelményei az adatbetöltés, feldolgozás, LSTM modell felépítése, tanítása, metrikák kiírása és eladási adatok előrejelzései voltak, valamint a felsorolt adatok mentése (18. Ábra Lásd: 34. oldal).



18. Ábra - A szoftver felépítése Forrás: Saját szerkesztés

Az adathalmaz a fentebb említett eladások összessége volt, ami 1 év eladását jelenti, a sell in és sell out értékek kilóban értendők. 1 sor az adathalmazból magába foglalja a termék nevét, az évet amikor az eladás megtörtént, a hetet amikor az eladás megtörtént, a termék kiszerezés grammban mérve, a listaárat, volt-e televíziós vagy online marketing a héten, a sell in adatokat (Spar, Tesco, Auchan) átlagolva, valamint a sell out adatokat (Spar, Tesco, Auchan) átlagolva. A sell in adatok kiegészítésre kerültek 4 lag oszloppal (így 1 adott sor tartalmazza az öt követő 4 hét sell in adatait is), valamint 1 sell in diff oszloppal ami az adott hét, és az azt követő hét közötti különbséget mutatja (5. Táblázat Lásd: 35. oldal).

Product Name	Year	Week Of The Year	Packaging Size_g	List Price	Marketing TV Campaign	Marketing Online Campaign	Sell in	Sell in_lag_1	Sell in_lag_2	Sell in_lag_3	Sell in_lag_4	Sell in_diff
Product 1	2022	44	425	1010	0	0	673,20	122,40	336,60	61,20	1193,40	550,80
Product 1	2022	45	425	1010	0	0	122,40	336,60	61,20	1193,40	153,00	-214,20
Product 1	2022	46	425	1010	0	0	336,60	61,20	1193,40	153,00	153,00	275,40
Product 1	2022	47	425	1010	0	0	61,20	1193,40	153,00	153,00	0,00	-1132,20
Product 1	2022	48	425	1010	0	0	1193,40	153,00	153,00	0,00	0,00	1040,40

5. Táblázat – Adathalmaz jellege LSTM formára hozás előtt Forrás: Saját szerkesztés

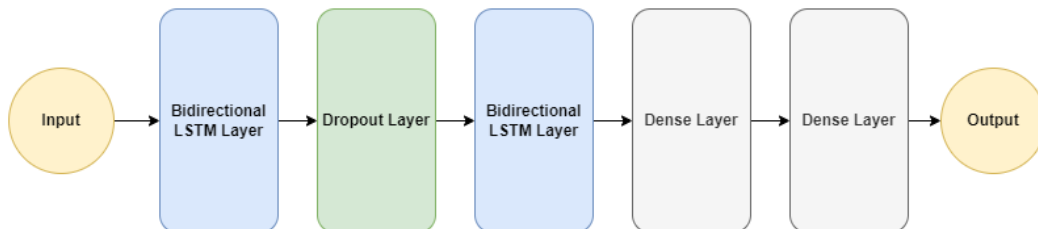
Az ötlet az volt mögött, hogy átlagoljuk azoknak a viszonteladóknak a sell in és sell out adatait, akik szolgáltatnak sell out információkat a vállalatnak, hiszen ezekből megállapítható a vásárlók vásárlási szokásainak intenzitása, amely által később sell out adatokat lehet generálni azokhoz a boltokhoz, amelyek nem nyújtanak sell out információkat.

A felhasznált technológia a pythonon belül, az adatfeldolgozásra pandas és numpy volt, a különböző megjelenítésekre és diagramokra pedig matplotlib-et használ a rendszer. Az adatok LSTM formájába való átalakításért a Tensorflow [21] felel, valamint magáért a modell felépítéséért a Keras technológia [22] nyújtotta eljárásokat használja a szoftver. A rendszer 6 nagyobb részből áll. Először is a lementett adatot kell betölteni, aztán jön az előfeldolgozás rész, ahol $]-1;1[$ határok közé skálázzuk le a sell in, sell in lag, sell in diff és sell out adatokat, valamint a Marketing TV és Online Campaign oszlopokat bináris adatokká alakítjuk. A Year, Week Of The Year, Packaging Size_g, List Price adatokat pedig pandas könyvtár get_dummies függvényének segítségével one hot encoding formára hozzuk. Az átalakítás után egy 3519 soros és 192 oszlopos adathalmazt kapunk (19. Ábra Lásd: 36. oldal). A sorokbeli csökkenést a lag oszlopok behozatala okozza, mivel ez a módszer azzal jár, hogy minden 53. sor végén (1 termék adatai) el kell dobunk 4 sornyi adatot, mivel NaN értékek jelennének meg az adathalmazban ha nem tennénk (több cikk is hivatkozik rá, hogy a teljesítménybeli növekedés nagyobb ennek a módszernek a használatával mint sem nélküle) [23] [23] .



19. Ábra - Adathalmaz jellege LSTM formára hozás után Forrás: Saját szerkesztés

Érdemes megemlíteni az LSTM architektúra kidolgozását és felépítését. Egy multi layer LSTM bizonyult a tesztek alapján a célravezetőnek, ami Bidirectional LSTM rétegeket használ. A felépítés sorrendje a következő: 1 réteg bidirectional LSTM, ezt 1 dropout követi (az overfitting elkerülése végett), még 1 réteg bidirectional LSTM, majd 2 dense réteg különböző számú neuronokkal (20. Ábra). A bidirectional LSTM használata a végzett teszt eredmények alapján indokolt volt, ezt a megelőző LSTM tesztek igazolták, ugyanis ebben az esetben a modell nem tudott rendesen feltanulni a szolgáltatott adatokon.



20. Ábra - LSTM modell rétegei Forrás: Saját szerkesztés

5.4 Valós idejű hírek feldolgozása RAG-on keresztül

A média által közölt hírek jelentős mértékben befolyásolják a fogyasztói szokásokat, ami meghatározza, hogy az előrejelzett forecast sikeres volt -e (tehát kellő hibahatáron belül esett), vagy sem. Így hiba lenne azt feltételezni, hogy elég kizárólag a múltbeli adatokra támaszkodva megpróbálni előre jelezni a jövőbeli eladásokat, mivel a képlet egy jelentős darabja hiányzik a hírek ismerete nélkül.

A Contextual Prompt segítségével lehetséges minden hír értékelése előtt automatizáltan utasítás sorozattal ellátni az LLM-et, amely tartalmaz minden fontos információt számára, mint például, hogy melyik cégnek dolgozik, mivel foglalkozik a cég, milyen szempontból kell kiértékelnie a híreket, mi a célja, milyen formátumú az elvárt válasz stb.

Ahhoz, hogy a feldolgozás minősége a lehető legmagasabb legyen, érdemes növelni az LLM számára elérhető információ halmazt, aminek az egyik módja a korábban tárgyalt RAG keretrendszer kialakítása. A vektor adatbázisban olyan szakirodalmak, kutatások, tudományos folyóiratok kerülnek eltárolásra, amelyek általános gazdasági tudással, FMCG termékekkel kapcsolatos tudással és média és fogyasztótársadalom viszonyával foglalkoznak, hogy a lehető legjobb minőségű feldolgozást nyújtsa az LLM.

A hírek beszerzése az internetről fog történni, hírportálok API-ján keresztül. A projekt szempontjából ez lehet a legcélravezetőbb, mivel ezek azok a források, amelyek a legnagyobb számban eljutnak a fogyasztó közösséghez, így hatással lehetnek annak a viselkedésére.

Ahhoz, hogy az LLM által kiértékelt hírek adatai bemenetet képezhessenek egy LSTM modell számára, ahhoz az LSTM modellnek korábban tanulnia kellett ilyen adatokon. Ehhez szükséges lesz egészen addig visszamenőleg feldolgozni a híreket, amikori dátumtól kezdődően az LSTM modell tanítása történik az adatokon. A kapott eredmények hozzáfűzésre kerülnek az eladási adatok adattáblájához, ezáltal a tanító adathalmazhoz.

6. Az eredmények

6.1 A tanítás folyamata

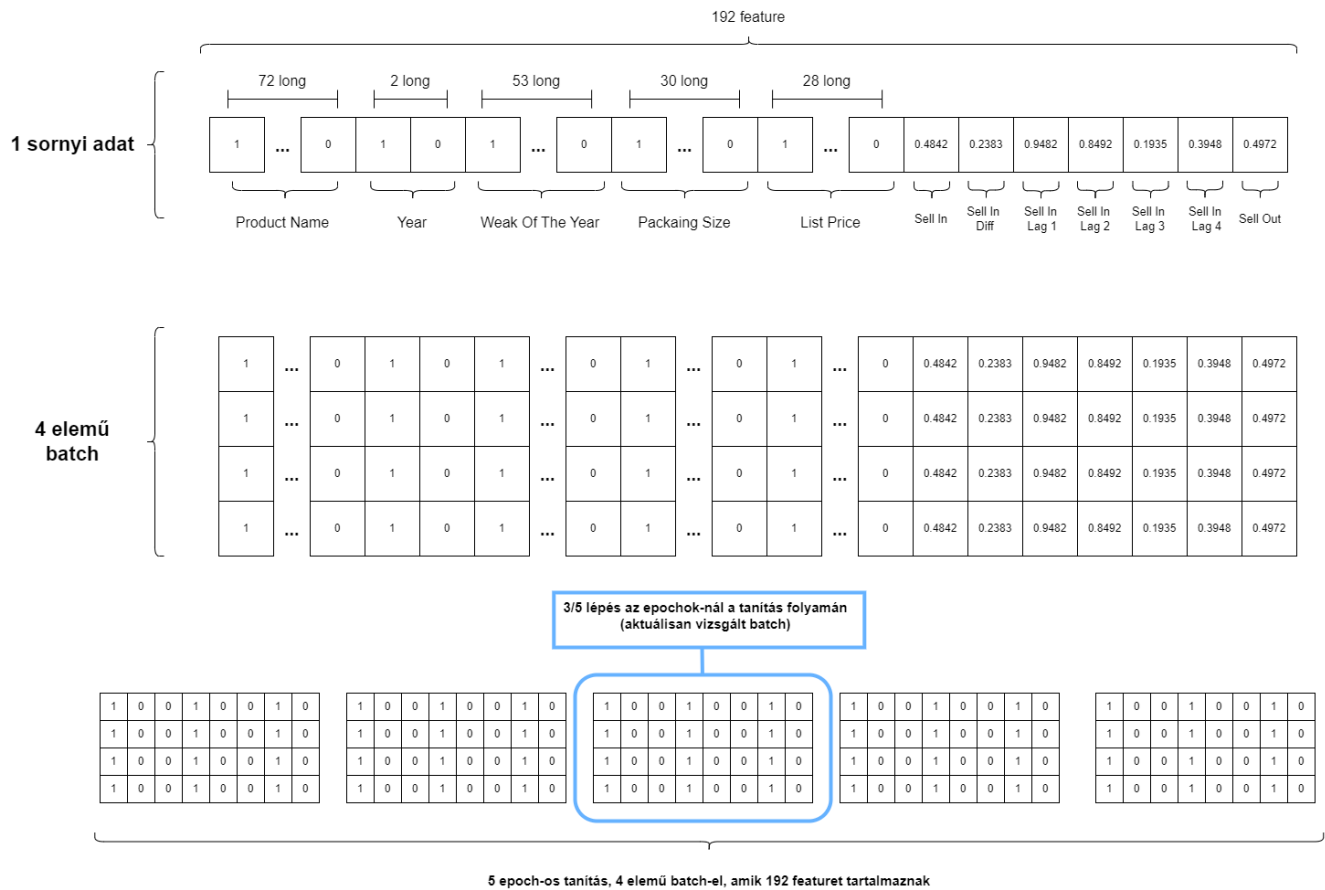
Az elkészült szoftver a fentebb leírtak szerint egy Jupyter Notebookbeli Python program, ami teljes terjedelmében 247 sort tesz ki. Ez tartalmazza az összes feldolgozási műveletet, az adatforrás betöltésétől, egészen az előrejelzés előállításáig. Az adatforrás neve „Training_Data.csv”, ez egy 3519 adatsort és 192 oszlopot tartalmazó adatforrás. Ez tartalmazza a fentebb említett termék adatokat. A szoftvernek az adatforrás ketté választása után (6. Táblázat) a sell out adatokat kell helyesen előre jelezni.

Adathalmaz neve	Adatok száma	Karakterisztikák	Magyarázat	Forrás
Tanítási adathalmaz	2815	192	Felelős a tanítás elvégzéséért	Előfeldolgozott vállalati adatok
Teszt és egyben validációs adathalmaz	704	192	Felelős az adatok teszteléséért	Előfeldolgozott vállalati adatok

6. Táblázat - Az adathalmazok jellege Forrás: Saját szerkesztés

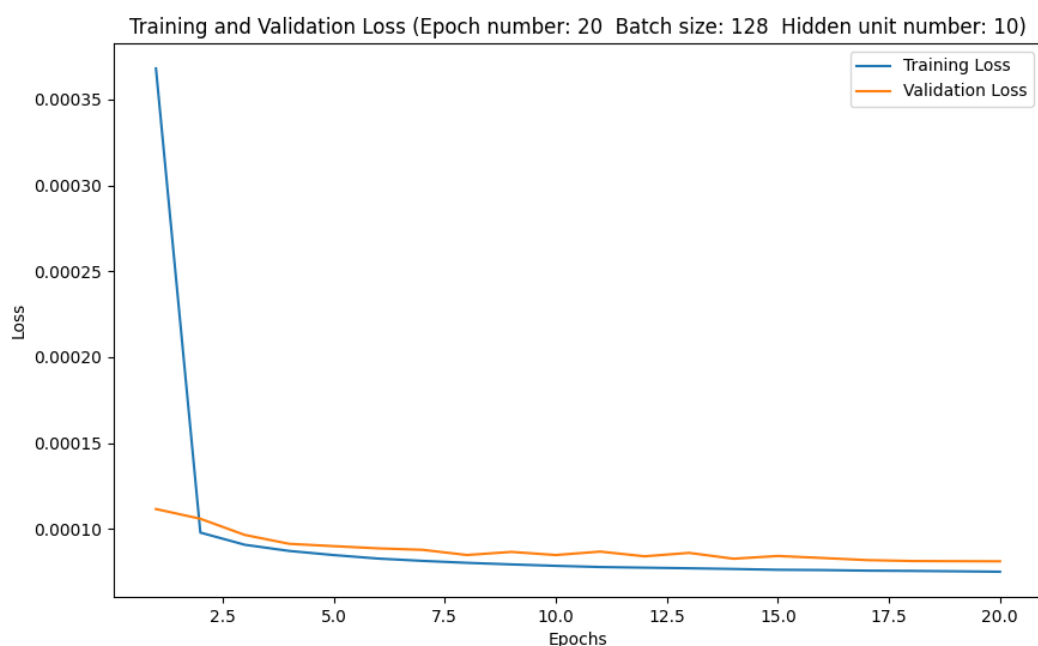
A tanítás fázis sokszor lefuttatásra került, hogy tesztelve legyen a modell számára optimálisnak mondható paraméterezés. A validációs és tanítási görbékre lett meghozva a döntés, hiszen a tanítás ezen szakaszában, ezek tükrözik a legjobban azt, hogy mennyire lesz pontos a későbbi előrejelzés. Minden ehhez szükséges grafikon a fentebb említettek szerint lementésre került. 5 féle paramétert állíthatunk különböző méretekre a tesztek során, ezek a batch size, tehát hogy mekkora adatcsomagok kerülnek feldolgozásra 1 epoch alatt. Az epoch numbert, ami azért felelt, hogy hányszor iteráljon át a tanítási adatbázison a szoftver, a hidden unit numbert (első és második réteg), ami pedig a rejtett karakterisztikák számát határozza meg, ezen kívül még az Adam optimizer tanulási gyorsaságát lehet paraméterezni a programon belül. Fontos volt, hogy az over fitting-et, és az under fitting-et kiküszöböljük a tanulás során, mivel a paraméterek hibás beállítása esetén mindkettő bekövetkezhet.

Karakterisztikából 192 van, ezek tartalmazzák a termék nevét, évet, hetet, kiszerelést, listaárat, sell in adatot (kilóban), 4 sell in lag-ot, 1 sell in diff-et és a sell out adatot (21. Ábra). Az adatok ebben a szakaszban már skálázva, binárisra alakítva, valamint one hot encodingolva vannak.



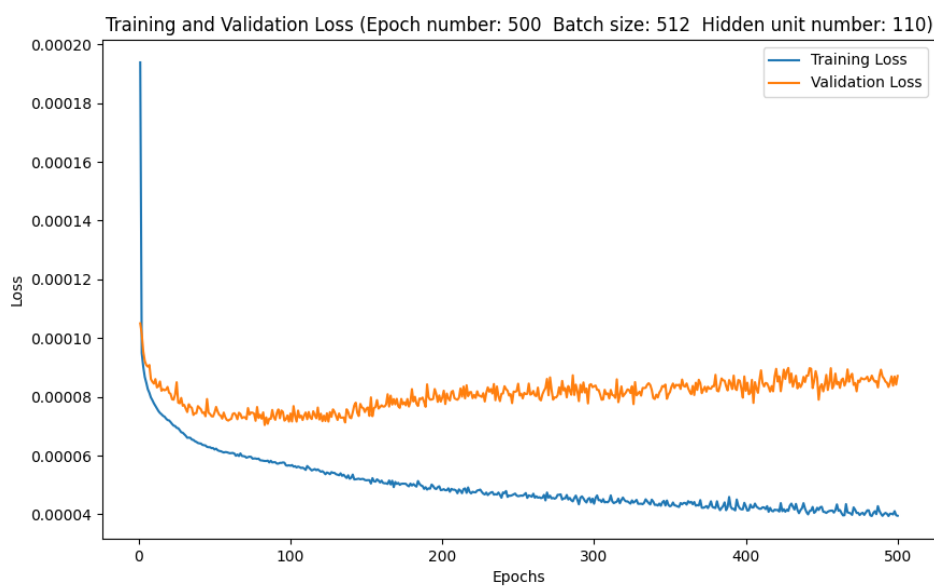
21. Ábra - Összefoglaló táblázat a feature, batch, és epoch fogalmak felépítéséről Forrás: Saját szerkesztés

Underfitnek azt hívjuk, amikor a modell túl egyszerű, és nem tud mintázatokat felismerni az adatok között, ilyenkor a tanítási és validációs görbe nagyon ellaposodik és egymáshoz ér (22. Ábra Lásd: 40. oldal).



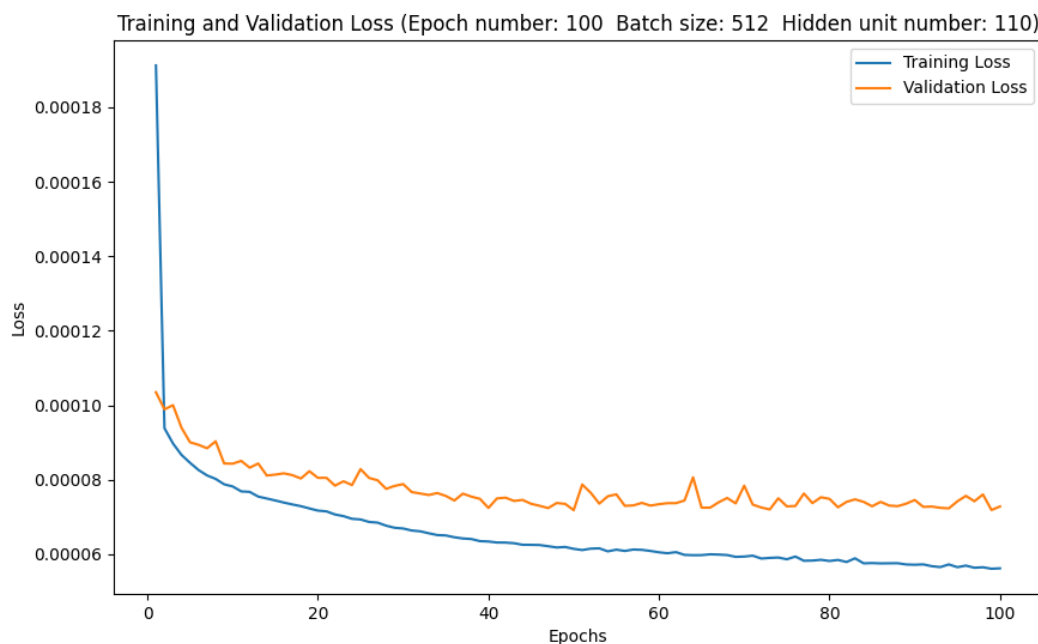
22. Ábra - Példa az underfitting esetére Forrás: Saját szerkesztés

Az overfitting esete pedig, amikor a modell túlságosan megtanulja az adathalmazt, és fals pozitív jeleket kezd el előrejelzni, tehát jó megoldásokat hallucinál a valósághoz képest, ilyenkor a tanítási és validációs görbe eltávolodik egymástól (23. Ábra).



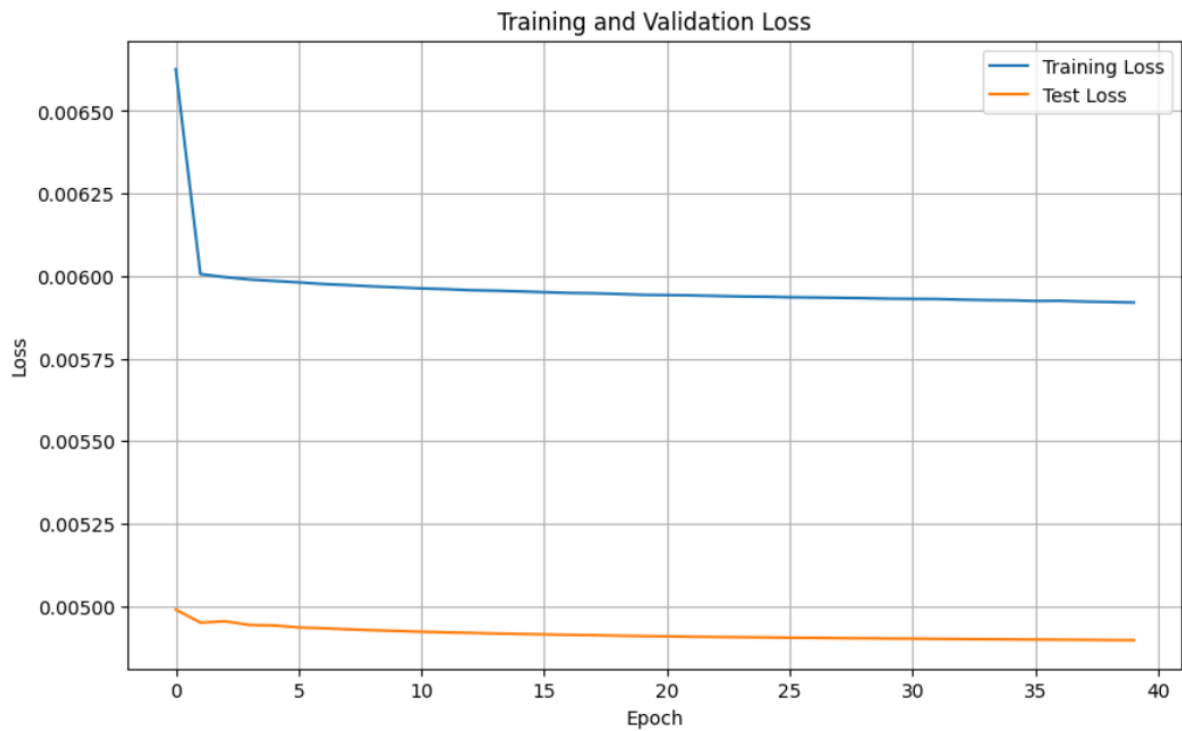
23. Ábra - Példa az overfitting esetére Forrás: Saját szerkesztés

Optimális paraméterezés esetén a tanítási a validációs görbe alatt halad, jól igazolva a helyes illeszkedés példáját (24. Ábra).



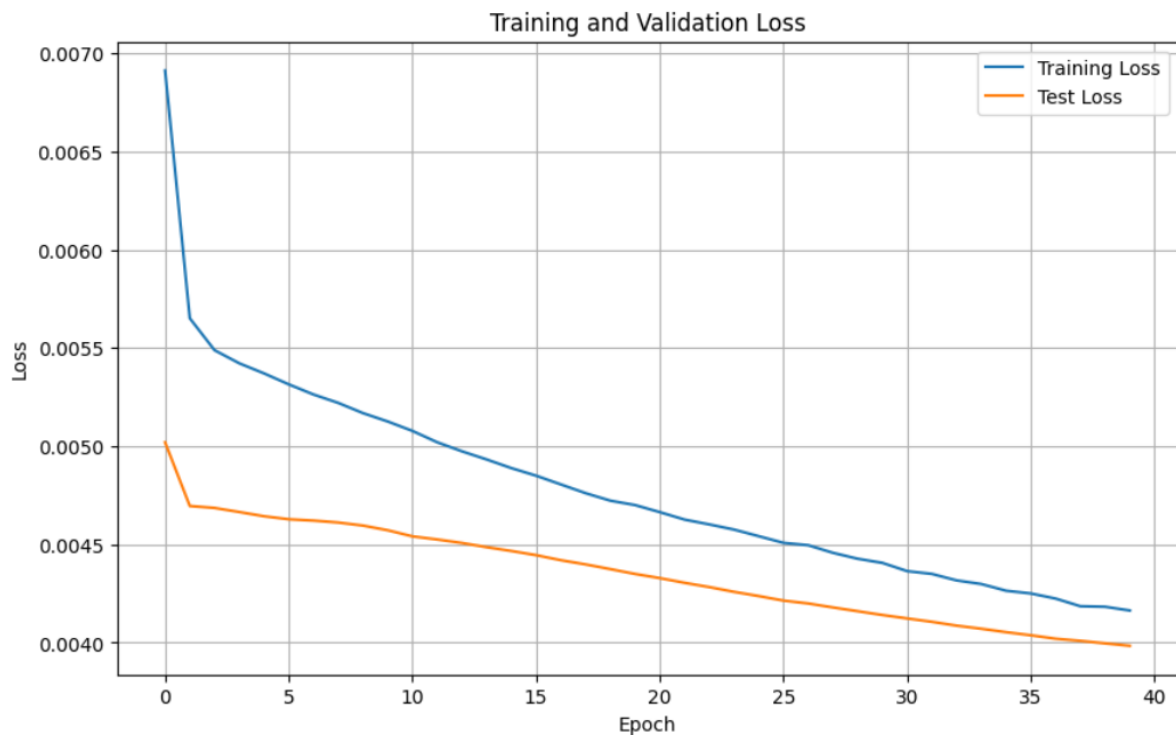
24. Ábra - Példa a helyes illeszkedésre Forrás: Saját szerkesztés

A tesztelés indokolta a bidirectional LSTM használatát, mivel enélkül nem tanulta meg az adatok komplexitását megfelelően a modell, ezt igazolja az alábbi diagram is (25. Ábra Lásd: 42. oldal).



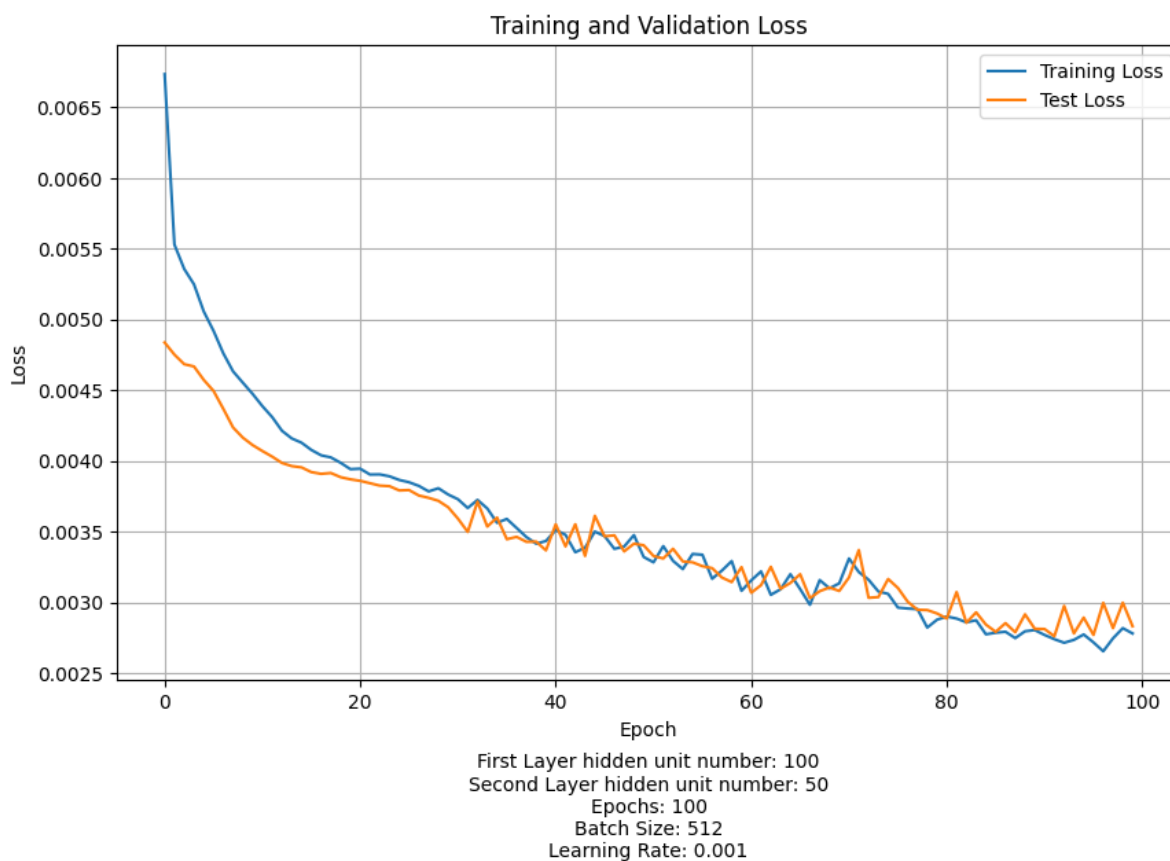
25. Ábra – Tanítási és validációs görbe viszonya *Directional LSTM* használatakor Forrás: Saját szerkesztés

A bidirectional LSTM rétegek bevezetése után szemmel látható javulás történik a 2 görbe egymáshoz való viszonyában (26. Ábra Lásd: 43. oldal). Ezután már csak a paramétereket kellett megfelelően optimalizálni.



26. Ábra Tanítási és validációs görbe viszonya Bidirectional LSTM használatakor Forrás: Saját szerkesztés

A mi esetünkben az optimum az első és második Bidirectional LSTM réteg hidden unit számát érintően az első rétegnél 100, a másodiknál 50-re esett. Az epoch number 100-ra, a batch size 512-re és az adam optimizer learning rate-je 0.001-re esett (27. Ábra Lásd: 44. oldal). A tanítási és validációs görbe kicsit zajos ebben az esetben, viszont a feltanulás a dolgozat ebben a fázisában elfogadhatónak mondható.



27. Ábra - Legjobb megtalált illeszkedés esete Forrás: Saját szerkesztés

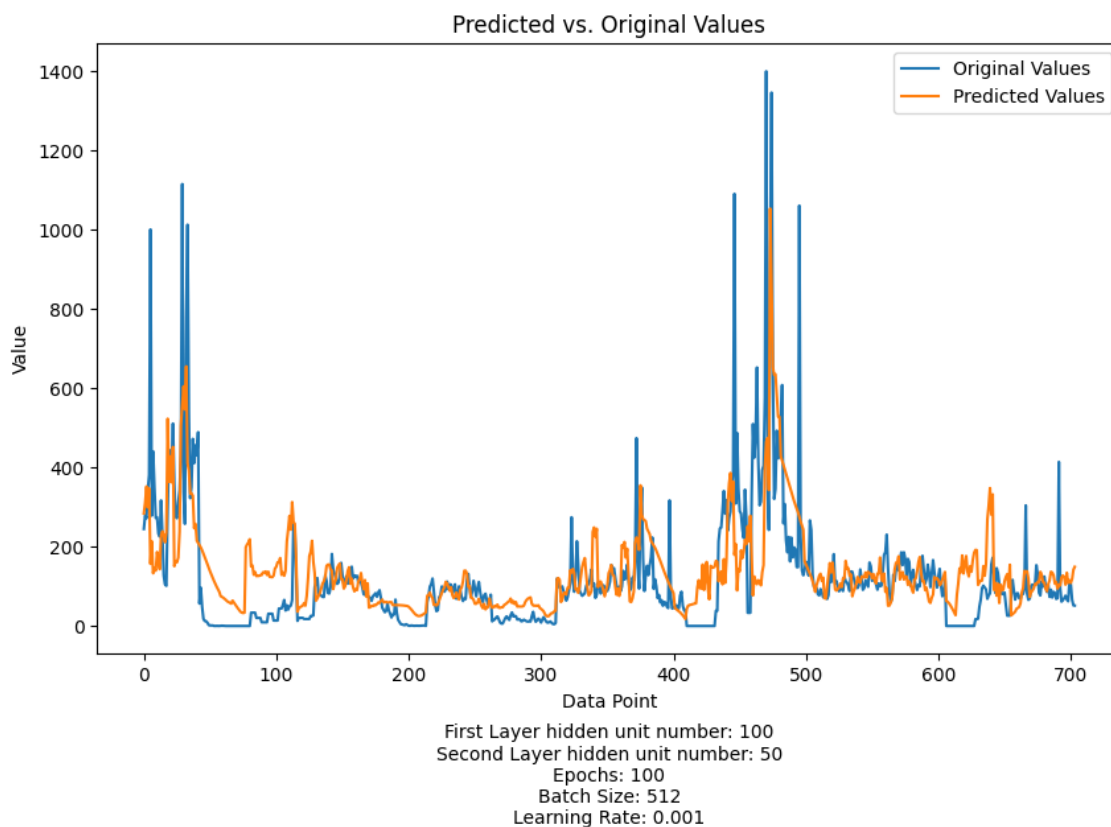
6.2 Az előrejelzés pontossága

Az előrejelzés pontosságának és elfogadhatóságának meghatározásához a MSE, RMSE és R^2 számokra támaszkodunk. Az optimum kiértékeléséhez több tesztet futtatunk, átírva magát az architektúrát és a hiperparamétereket. A fentebb említett bidirectional LSTM használata nagyban hozzásegített a pontosabb eredmények eléréséhez, a legjobb paraméterezés zöld színnel kiemelve látható a táblázatban (7. Táblázat Lásd: 45. oldal).

Layer Number	LSTM Layer Type	First Layer Hidden Unit Number	Second Layer Hidden Unit Number	Epoch Number	Batch Size Number	Learning Rate	MAE	RMSE	R ²
1	Directional LSTM	100	0	40	128	0.001	89,47	158,46	0,13
1	Directional LSTM	100	0	40	128	0.0001	89,01	158,28	0,15
2	Directional LSTM	100	50	40	128	0.0001	93,68	160,01	0,08
2	Bidirectional LSTM	100	100	40	128	0.0001	77,08	140,99	0,20
2	Bidirectional LSTM	100	50	100	256	0.0001	78,06	138,20	0,23
2	Bidirectional LSTM	100	50	70	256	0.001	74,94	131,78	0,30
2	Bidirectional LSTM	100	50	100	256	0.001	77,97	123,90	0,38
2	Bidirectional LSTM	100	50	100	512	0.0001	79,27	139,31	0,22
2	Bidirectional LSTM	100	50	100	512	0.001	73,57	120,27	0,42
2	Bidirectional LSTM	100	50	150	512	0.001	81,93	133,20	0,29
2	Bidirectional LSTM	150	100	100	512	0.001	82,83	131,31	0,31
2	Bidirectional LSTM	100	50	90	512	0.001	73,32	124,75	0,37

7. Táblázat - Teszt esetek paramétereit és mérőszámai Forrás: Saját szerkesztés

A mérőszámok mellett összevetettük az előrejelzett adatokat a valós sell out adatokkal, ezáltal láthatóvá vált, hogy milyen módon tudja lekövetni a modellunk az adatokat (28. Ábra Lásd:46. oldal). A jelenlegi legjobb paraméterezés szerint jól látható, hogy a modell a trendeket szépen lekövetni, viszont a pontos méretek megbecsléséhez tovább dimenziók bevezetésére lesz szükség (például a hírek elemzése, vagy egyéb publikus a Központi Statisztikai Hivatalból származó adatok hozzárendelése).



28. Ábra - Valós és előrejelzett adatok optimális paraméterek esetén Forrás: Saját szerkesztés

Az előrejelzett adatokat mélyebben elemezve, látszik, hogy átlagosan 73,57 kg egy héten a becsült adatok és a valós között a tévedés (ez éves szinten akár 3500 kg körüli is lehet). Ez az adatok közötti nagy eltéréseknek köszönhető, ennek kiküszöbölésére szolgál majd a további dimenziók beépítése, viszont a modell a hetek közötti trendeket már ebben a fázisban is szépen leírja, ez segítségül szolgálhat 1 termék eladási trendjeinek előrejelzéséhez.

7. Konklúzió

Mivel a piac hektikus változásainak mértéke várhatóan nem fog csökkenni, hanem szinten marad, sőt, nőni fog, így a vállalat példáján keresztül kijelenthető, hogy egy alternatív tervet engedélyező stratégia megoldást jelenthet a bevezetésben ismertett gazdasági problémára.

Az előző TDK alatt elért eredmények alátámasztják azt az elképzelést, hogy egy gépi tanulási 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 aktuális 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 RAG rendszer által. A jelenlegi előrejelzés hibamértékét előre láthatólag csökkenteni fogja, hogyha egy diverzifikáltabb képet kaphatunk a piac működéséről a hírek, globális adatok és a belső komplexebb vállalati adatok gyarapításával.

Az előrejelzett adatokat már fel lehet használni tervezéshez, mivel az általunk fejlesztett rendszer az eladási trendeket szépen leírja, ami (az eladás mértékétől eltekintve) segítséget nyújthat a hétről hétre való kalkulációban.

A Bidirectional LSTM használata nagyban megnövelte a feltanulás és előrejelzés minőségét is, azonban a mérőszámok azt mutatják, hogy lehetőség van még a modell finomítására, ami így potenciálisan akár sokszorosán jobb előrejelzéseket is nyújthat a jövőben. A paraméterek finomításával a rendszer képes lesz akár hetekre előre mind az eladási trendeket, mind az eladások mértékét egy megbízható hibahatáron belül meghatározni.

7.1 Jövőbeli munka

A tanulmány következő lépcsőfoka a sell out adatok további előállítása és pontosítása lesz, majd az ezeken való modell tanítása, illetve a RAG rendszer vektor adatbázisának kiépítése, továbbá a kommunikáció megvalósítása az LLM, vektor adatbázis és felhasználó között.

Ez a jövőbeli rendszer együttes egy alapvető alkotó eleme lesz egy komplett, alternatív stratégiákat alkotó DTR-nek a vállalaton belül. Ez a kész program rendelkezni fog az LSTM/BiLSTM modell súlyozás és eltolás értékeivel, az inputjai pedig az aktuális gazdasági értékek lesznek, mint például az előző havi valós sales adatok, illetve a RAG rendszer által kiértékelt hírek adatai. Ez a DTR az előrejelzéseket felhasználva jelzi a meglévő stratégiai tervek közötti váltás szükségességét, ezzel segítve a céget a piaci mozgás agilis lekövetésében, továbbá konkrét stratégiai terveket is előállít a vállalat számára, amikre áttérhetnek. Ennek a DTR-nek a technikai működése még a kutatás jövőbeli részét képezi.

A jelenlegi projekt az alapját képezi egy másik rendszernek is, amely hasonlóan a forecast prediktáláshoz, úgyszint eredmények előrejelzését célozza meg. Ez a rendszer egy akciótervező rendszer, amely azt prognosztizálja a vállalatnak, hogy milyen eladásra számíthatnak egy értékesítési csatornán egy termék esetén, ha bevezetnek egy bizonyos akciót, például 2+1-et, vagy egy hagyományos 20%-os akciót bizonyos időre. Jelenleg a cégnek nincs megbízható forrása ezeket az információkat illetően, így nem tudnak megfelelően tervezni ezekkel az akciókkal, így előfordulhat, hogy egy áruházlánc boltjában egy adott termékből nincs annyi, mint amennyi eladás történne, ha lenne. Gazdaságilag ez nagy rizikót hordoz magában, mert azon túl, hogy nem történik eladás, annak is fennáll az esélye, hogy a vásárló a cég hiányzó terméke helyett egy konkurencia márkáját választja, így lehetséges, hogy a másik termék rendszeres vásárlója lesz a mi cégünk terméke helyett.

Ez a rendszer hasonlóan a forecast előrejelzőhöz, úgyszint gépi tanulós módszereket fog alkalmazni a múltbeli akciók trendjeinek megtanulására, inputparamétere pedig a bevezetendő akció típusa, mértéke és időintervalluma lesz.

A forecast-előrejelző és akciótervező rendszer kritikus továbbfejlesztései közé tartozik még, hogy ne csak egy terméken, hanem egy divízió minden termékén működjön. A kutatás jövőjében a vállalattal együttműködve, a rendszerek kiterjesztésre kerülnek a gabonapelyhek divíziójáról azokra a további divíziókra, melyek adatainak jellege ezt lehetővé teszi. Ezen kívül, a nagyon távoli jövőben, igény esetén a rendszereket a klaszteren belüli országok vállalatainak képviselőivel együttműködve továbbfejlesztésre kerülnek, hogy az ottani igényeknek megfelelően is működhessenek.

Ezek a rendszerek jellegükből adódóan folyamatos karbantartást és fejlesztést igényelnek, hiszen egy olyan rendszer, amelyet utoljára több hónappal vagy évvel ezelőtti adatokkal

tanítottak, nem lesz képes az aktuális piaci igényeknek megfelelően működni, így a kutatás jövőbeli részét képezi a rendszer tanításának valamiféle automatizált megoldása, melynek technikai részleteire még nem került sor a kutatás folyamán.

Ahhoz, hogy a cég dolgozói ezeket a szoftvereket megfelelően képesek legyenek használni a mindennapos munka folyamán, szükséges fejleszteni egy graphical user interface (GUI)-t is hozzá, amely áthidalja a felhasználó és a gép közötti kommunikáció nehézségeit. Ennek kellően egyértelműnek kell lennie ahhoz, hogy az olyan felhasználók is magabiztosan tudják használni, akiknek nem céljuk megérteni a szoftver mögöttes működését.

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 hálózata architektúráról Forrás: [3] nyomán saját szerkesztés	11
4. Ábra - Illusztráció egy rekurrens neurális hálózata architektúráról Forrás: [9] nyomán saját szerkesztés	13
5. Ábra - Az LSTM cella felépítése Forrás: [11] nyomán saját szerkesztés	15
6. Ábra - Kaszkádosított cellák Forrás: [11] nyomán saját szerkesztés	17
7. Ábra - A BiLSTM működése. Forrás: Saját szerkesztés	18
8. Ábra - A mondat feldolgozása hagyományos LSTM segítségével. Forrás: Saját szerkesztés	19
9. Ábra - A BiLSTM működése nyelvi feldolgozásnál. Forrás: Saját szerkesztés	20
10. Ábra - Az LLM-ek működésének ábrázolása Forrás: Saját szerkesztés	23
11. Ábra - A Mistral7B teljesítménye diagramon ábrázolva egyéb state of the art modellekkel szemben. Forrás [16]	24
12. Ábra - A RAG rendszer általános működésének bemutatása Forrás: Saját szerkesztés	26
13. Ábra - A prototípus szoftver valós és előrejelzett adatai egymáshoz viszonyítva Forrás: Saját szerkesztés	27
14. Ábra - Auchan eladások Kullback-Leibler [19] eloszlása Forrás: Saját szerkesztés	31
15. Ábra – Spar eladások Kullback-Leibler [19] eloszlása Forrás: Saját szerkesztés	31
16. Ábra– Tesco eladások Kullback-Leibler [19] eloszlása Forrás: Saját szerkesztés	31
17. Ábra– Boltok Kullback-Leibler [19] normális eloszlása Forrás: Saját szerkesztés	32
18. Ábra - A szoftver felépítése Forrás: Saját szerkesztés	34
19. Ábra - Adathalmaz jellege LSTM formára hozás után Forrás: Saját szerkesztés	36
20. Ábra - LSTM modell rétegei Forrás: Saját szerkesztés	36
21. Ábra - Összefoglaló táblázat a feature, batch, és epoch fogalmak felépítéséről Forrás: Saját szerkesztés	39
22. Ábra - Példa az underfitting esetére Forrás: Saját szerkesztés	40
23. Ábra - Példa az overfitting esetére Forrás: Saját szerkesztés	40
24. Ábra - Példa a helyes illeszkedésre Forrás: Saját szerkesztés	41

25. Ábra – Tanítási és validációs görbe viszonya Directional LSTM használatakor Forrás: Saját szerkesztés	42
26. Ábra Tanítási és validációs görbe viszonya Bidirectional LSTM használatakor Forrás: Saját szerkesztés	43
27. Ábra - Legjobb megtalált illeszkedés esete Forrás: Saját szerkesztés.....	44
28. Ábra - Valós és előrejelzett adatok optimális paraméterek esetén Forrás: Saját szerkesztés	46

TÁBLÁZATJEGYZÉK

1. Táblázat - Az OHE bemutatása Forrás: Saját szerkesztés.....	21
2. Táblázat - A vector embedding bemutatása Forrás: Saját szerkesztés	22
3. Táblázat - A sell in adatok (részlet az adathalmazból) Forrás: Saját szerkesztés	30
4. Táblázat - A sell out adatok (részlet az adathalmazból) Forrás: Saját szerkesztés	30
5. Táblázat – Adathalmaz jellege LSTM formára hozás előtt Forrás: Saját szerkesztés	35
6. Táblázat - Az adathalmazok jellege Forrás: Saját szerkesztés	38
7. Táblázat - Teszt esetek paramétereit és mérőszámai Forrás: Saját szerkesztés	45

IRODALOMJEGYZÉK

- [1] J. Schmidhuber, “Annotated History of Modern AI and Deep Learning.” arXiv, Dec. 29, 2022. 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. Tallec and Y. Ollivier, “Unbiasing Truncated Backpropagation Through Time.” arXiv, May 23, 2017. 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.” arXiv, Aug. 01, 2023. doi: 10.48550/arXiv.1706.03762.
- [14] A. Q. Jiang *et al.*, “Mistral 7B.” arXiv, Oct. 10, 2023. doi: 10.48550/arXiv.2310.06825.
- [15] P. Lewis *et al.*, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.” arXiv, Apr. 12, 2021. doi: 10.48550/arXiv.2005.11401.
- [16] G. van Rossum, “Python reference manual,” Art. no. R 9525, Jan. 1995, Accessed: Apr. 02, 2024. [Online]. Available: <https://ir.cwi.nl/pub/5008>
- [17] J. Reback *et al.*, “pandas-dev/pandas: Pandas 1.0.0.” Zenodo, Jan. 29, 2020. doi: 10.5281/zenodo.3630805.
- [18] C. R. Harris *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.
- [19] 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.

- [20] “An Introduction to Statistical Learning,” An Introduction to Statistical Learning. Accessed: Apr. 02, 2024. [Online]. Available: <https://www.statlearning.com>
- [21] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.” 2015. Accessed: Nov. 07, 2023. [Online]. Available: <http://download.tensorflow.org/paper/whitepaper2015.pdf>
- [22] “keras-team/keras.” Keras, Apr. 02, 2024. Accessed: Apr. 02, 2024. [Online]. Available: <https://github.com/keras-team/keras>
- [23] D. Lengacher, “LSTMs with Lagged Data,” Medium. Accessed: Apr. 02, 2024. [Online]. Available: <https://medium.com/@dclengacher/lstms-with-lagged-data-cc03a3a8cfcf>