Miskolci egyetem

Gépészmérnöki és Informatikai Kar

Általános Informatikai Intézeti Tanszék


# Vállalati chatbot alkalmazás fejlesztése

# Szakdolgozat

**Készítette:**

**Név:** Csomor Bence Patrik

**Neptunkód:** TVIK4I

**Szak:** Mérnökinformatikus BSc

Korszerű web technológiák szakirány

# EREDETISÉGI NYILATKOZAT

Alulírott ……Csomor Bence Patrik.……..; Neptun-kód:……TVIK4I…

a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős ……mérnökinformatikus alapszakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy

……Vállalati chatbot alkalmazás fejlesztése……

című szakdolgozatom/diplomatervem saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc,...2023.....év ……06……..hó ……02..nap

…….……..…

Hallgató

**Miskolci Egyetem**
**Gépészmérnöki és Informatikai Kar**

**Általános Informatikai Intézeti Tanszék**
3515 Miskolc-Egyetemváros

Szak**: Mérnökinformatikus**

Szakirány: Korszerű web technológiák

Szakdolgozat azonosító:
**IAL/TVIK4I/BSc/2023**
**Intézmény azonosító: FI 87515**

# S Z A K D O L G O Z A T  F E L A D A T

## CSOMOR BENCE PATRIK

BSc mérnökinformatikus jelölt részére

A tervezés tárgyköre: **szoftverfejlesztés**

A feladat címe: Vállalati chatbot alkalmazás fejlesztése

**A feladat részletezése:**

- A mesterséges intelligencia elméleti hátterének feldolgozása
- Programspecifikáció kidolgozása és fejlesztő eszközök ismertetése
- A program megvalósításához szükséges feltételek kidolgozása
- Chatbot integrálása a webalkalmazásba

| Tervezésvezető: | Tanszék, beosztás: |
|---|---|
| Kunné Dr. Tamás Judit | Informatikai Intézet, Egyetemi Adjunktus |
| Konzulens(ek): | Cég, beosztás: |
| Dézsi István | Bosch, Csoportvezető |

A szakdolgozat kiadásának időpontja: 2023.02.20.

A szakdolgozat beadásának határideje: 2023.05.19.

Miskolc, 2023.03.06.

**Prof. Dr. Kovács László**

tanszékvezető egyetemi tanár

1. A szakmai gyakorlat helye: _____

2. A szakmai gyakorlat vezetőjének neve: _____

3. A szakdolgozat módosítása: szükséges (a módosítást külön lap tartalmazza)
   nem szükséges (a megfelelő rész aláhúzandó)

   Miskolc, _____

   tervezésvezető aláírása

4. A tervezést ellenőriztem: (1) _____

   (2) _____

   (3) _____

   (4) _____

   dátum, tervezésvezető aláírása

5. A szakdolgozat beadható
   nem adható be

   Miskolc, _____

   konzulens aláírása tervezésvezető aláírása

6. A szakdolgozat ….. szövegoldalt,
   ….. db rajzot,
   ….. db CD mellékletet
   ….. egyéb mellékletet tartalmaz.

7. A szakdolgozat bírálatra: bocsátható

   nem bocsátható

   A bíráló neve, címe: _____

   Miskolc, _____

   tanszékvezető aláírása

8. Osztályzat: a bíráló javaslata: _____

   a tanszék javaslata: _____

   a Záróvizsga Bizottság döntése: _____

   Miskolc, _____

   a Záróvizsga Bizottság elnökének aláírása

# Tartalomjegyzék

# 1 Bevezetés

Már a Miskolci Egyetemi tanulmányaim során is érdekeltek az olyan alkalmazások, melyek megkönnyítik, esetenként élvezetesebbé tesznek bizonyos tevékenységeket. Mindig is lenyűgözött a technológia rohamos fejlődése és az utóbbi évek során felfigyeltem a mesterséges intelligencia térhódítására is. Manapság mindenhol az AI-ról hall az ember és nem véletlenül. Hihetetlen iramban fejlődik ez a technológia és még messze nem érte el a potenciáljának határait. Ma már az AI-Art és a különböző arcfelismerő szoftvereké a reflektorfény. Az én szakdolgozatom témája ezen érdeklődési körből származik. Mindig is érdekelt a már-már misztikusnak ható összetétele a mesterséges intelligenciáknak. A Mesterséges Neurális Hálókról szerettem volna gyarapítani a tudásomat ebben a projektben. A projektem egy, a Bosch köreiben fejlesztett, vállalati chatbot alkalmazás fejlesztése, amelyet a méltán híres *ChatGPT* webalkalmazás ihletett. Ezen alkalmazás a cég régi és új dolgozóinak egyaránt segítséget fog nyújtani az általános ügyek intézésében, melyeket még nem hajtott végre az adott dolgozó vagy már olyan rég volt ehhez hasonló feladata, hogy fölmerült az adott témával kapcsolatban pár kérdése. A projekt ötlete akkor merült fel mikor a főnökömmel beszéltük, hogy van egy chatbot alkalmazása a cégnek, de nem elégíti ki a felhasználói igényeket és a megvalósítása sem a legmegfelelőbb. Ezen a ponton született meg a vállalati chatbot fejlesztésének ötlete.

A projektem célja, az összes munkatárs által elérhető információs chatbot megalkotása. A programnak képesnek kell lennie a munkatárs által bevitt kérdésre kielégítő választ adnia, a kérdés témakörével kapcsolatban, oly módon, hogy felismerje a kérdés szövegkörnyezetét és egy erre betanított Mesterséges Neurális Háló segítségével döntést hoz a válaszát illetően.

Meghatározott témakörökkel kapcsolatban tudnia kell megfelelő választ adnia. Olyan témával kapcsolatban, amelyet még nem érintett a betanulás során vagy nincs köze egyetlen releváns témakörhöz sem, arra egy alapértelmezett és egyértelmű választ ad, miszerint nem érti a feltett kérdést, ezáltal megadva a lehetőséget, a kérdés átfogalmazására vagy más téma érintésére.

Ezt a szoftvert egy webalkalmazáson keresztül valósítom meg, amellyel minden munkatárs el fogja tudni érni egy link segítségével.

Szakdolgozatomban bemutatom a Bosch környezetet és a gyártási területet, amiben a Miskolci telephely foglalkozik. Bemutatásra kerül a chatbot fogalma, technológiái és története is. Ezt követően az általam elkészített webalkalmazás tervezési, megvalósítási és tesztelési lépései kerülnek bemutatásra.

## 1.1 Bosch bemutatása

Robert Bosch alapította meg "Finommechanikai és Elektrotechnikai Műhelyét" Stuttgartban 1886-ban, ami később a világszerte tevékenykedő Robert Bosch GmbH vállalattá vált. A cég történetét az innováció és a társadalmi elkötelezettség jellemezte.

### 1.1.1 Robert Bosch Magyarország

A Bosch csoport magyarországi története 1991-ben kezdődött a Robert Bosch Kft. megalapításával. Azóta számos telephely jött létre az ország különböző pontjain, többek között Budapesten, Hatvanban, Makláron és Miskolcon.

### 1.1.2 Robert Bosch Energy and Body Systems Kft. (RBHM)

A miskolci telephely 2002-ben kezdte meg működését először Robert Bosch Power Tool Elektromos Szerszámgyártó Kft. néven, majd 2003-ban megalakult a Robert Bosch Energy and Body Systems Kft. is.


*1. ábra: RBHM épülete, Miskolc [1]*

A Robert Bosch Energy and Body Systems Kft. (RBHM) 2003-ban alakult Miskolcon, kezdetben autóipari alkatrészek gyártására fókuszálva, mint például

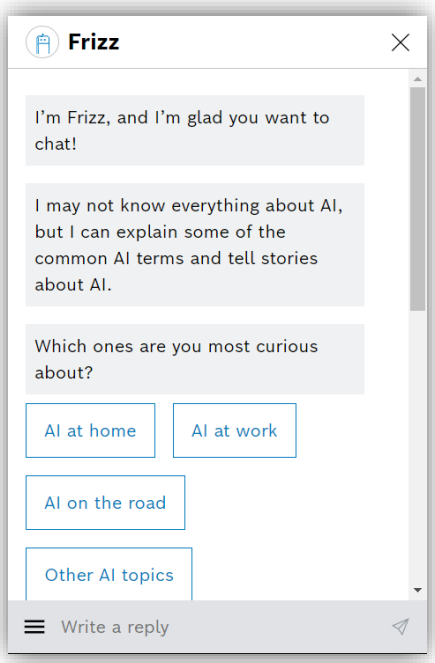
klímaberendezések, fékrásegítő motorok és menetstabilizátorok. Azóta több mint 200 millió autóipari termék került legyártásra. A legnagyobb sikerüket azonban az eBike meghajtás területén érték el, amely 2020-tól önálló üzletággá vált. Az eBike egy fejlett erőrásegítő rendszer, amely segíti a kerékpár hajtását különböző szenzorok segítségével. Miskolcon gyártják a meghajtóegységet, az akkumulátorokat és a teljes eBike rendszert, amely magában foglalja a meghajtóegységet, az akkumulátort, a kijelzőt és a szükséges kábeleket. [1]

### 1.1.3 Frizz chatbot

A Bosch vállalaton belül fejlesztettek egy chatbotot, amely a Frizz névre hallgat. A Frizz célja, hogy beszélgetéseket folytasson a felhasználókkal és információt nyújtson az mesterséges intelligenciáról (MI). A chatbot arra törekszik, hogy könnyen érthető és humoros módon mutassa be az MI bonyolult témáját.

A Frizz fókuszában a Bosch által alkalmazott MI témakörei állnak, amelyeket 2025-re az összes megoldásukba és termékükbe beépítenek, vagy ezen technológia segítségével fejlesztenek és gyártanak. Ide tartozik a hálózatba kapcsolt életre vonatkozó technológia, például okos sütők és intelligens algoritmusok. A Frizz interaktív és szórakoztató megközelítést alkalmaz annak érdekében, hogy bevonja a felhasználókat és bemutassa az MI különböző aspektusait azoknak, akik érdeklődnek iránta.

Frizz használatakor a felhasználók különböző irányokba terelhetik a beszélgetést a válaszok alapján. A Frizz kattintható válaszokat kínál, de a felhasználók hagyományos szöveges bevitellel is kommunikálhatnak a chatbottal. Célja, hogy rövid történetek és időnkénti rejtvények segítségével könnyen megérthetővé és szórakoztatóvá tegye az MI témáját.




*2. ábra: Frizz chatbot*

Az adatvédelem tekintetében a Frizz anonimizált módon tárolja az adatokat. A Frizz-chatet használó felhasználóknak nincs profiljuk, és az IP-címüket sem dolgozza fel. Amikor egy beszélgetés a Frizzel elindul, két sütit helyeznek el. Az egyik a chat alkalmazás futtatásához szükséges, míg a másik a kommunikáció megfelelő feldolgozását biztosítja. Ezek a sütik a felhasználó böngészőjében tárolódnak, és automatikusan törlődnek, amikor bezárják a weboldalt. Ezért nem lehetséges következtetéseket levonni az egyéni felhasználókról. Néhány használati analitikai adatot kinyernek a beszélgetésekből és a Bosch szerverein tárolják. Ide tartozik például a munkamenetek száma és időtartama, vagy hogy volt-e több interakció a chatben, gombokra kattintással vagy szöveges bevitellel, valamint a beírt szövegek. Bármelyikük között személyes adatokkal kapcsolatos bejegyzések találhatók, azokat egy Bosch munkatárs azonnal törli.

A Frizz a gépi tanulásra és a természetes nyelvfeldolgozásra (NLP) épül. A szabályalapú chatbotokkal ellentétben a Frizz algoritmusa lehetővé teszi a természetes nyelv megértését és a rá irányuló kérdések megválaszolását az adatbázisában szereplő általános információk alapján. Az adatbázisában főként a Bosch MI alkalmazásokhoz kapcsolódó témák találhatók. [2]

# 2 Elméleti háttér ismertetése

A következőkben bemutatom a chatbot fogalmát, elérhető változatait és nyelvi modelljeinek fajtáit, megvalósítási típusait. Ezt követően szó lesz részletesen a természetes nyelvi feldolgozásról (NLP). Kifejtem többek között a mesterséges neurális háló fogalmát, magát a mesterséges intelligenciát és kitérek két webfejlesztésben használt keretrendszerre is.

## 2.1.1 Chatbot ismertetése

A Chatbotok már évek óta léteznek, viszont csak az utóbbi években lett felkapott a felhasználók és a vállalatok körében. Sokféle változata létezik és a felhasználásuk is szerteágazó a világban.

A Chatbot egy olyan szoftver, amely képes megérteni, feldolgozni és reagálni a felhasználóra, ezzel olyan érzetet keltve benne, mintha egy másik emberrel kommunikálna. Összességében egy Mesterséges intelligenciával felruházott program, ami képes „csevegni” velünk. A felhasználási módjai közé tartozik a csevegés, szolgálhat információval, foglalhat nekünk szállást és még olyan dolgokra is fel lehet használni, melyek törvénybe ütköznek. Sajnálatos módon káros oldala is van ezeknek a programoknak, mivel a számítógépes vírusok terjedésének megkönnyítése mellett a közösségi hálókon is előfordul a használata, mint mesterségesen generált nézettség és/vagy követőtábor generálása. Személyi asszisztensként is remek segítséget tud nyújtani, továbbá az üzleti folyamatokat is nagyban fel tudja gyorsítani egy megfelelően kifejlesztett Chatbot, csak hogy említsek még egy pár felhasználási területet.

Az első Chatbotot egy az MIT-n dolgozó professzornak, Joseph Weizenbaumnak köszönhetjük. Ez a program az ELIZA névre hallgat, melyet az 1960-as években adott ki. ELIZA-t követte számos többé-kevésbé ismert alkalmazás is: [3]

- PARRY
- Jabberwacky
- A.L.I.C.E
- Siri
- Google Assistant
- Cortana
- Alexa

A legjelentősebb változata manapság egyértelműen a ChatGPT (Generative Pre-trained Transformer) nevezetű, nagy méretű nyelvi modell, amit az OpenAI nevű cég adott ki 2021-ben. Ezt úgy tervezték, hogy a felhasználó segítségére legyen, ember szerű szöveget generálva. A társalgás generálás mellett nyelvi fordításra is jól használható. Ez a modell képes nyelvtanilag gördülékeny és látszólag emberi válaszokat adni különféle kérdésekre változatos témákban, mint a chatbotok nagy része, ez a modell sem tökéletes. Mivel tervezéséből adódóan megpróbál minden kérdésre a lehető legkielégítőbb választ adni, annak ellenére is, hogy egy elképesztően nagy adatbázison lett betanítva, ez sem tud mindenre megfelelő választ adni. Ebből kifolyólag hajlamos a modell hibás válaszokat generálni vagy pontatlan információkkal szolgálni. A modell pontossága 79%-os érték körül mozog. Nyelvezete illedelmes, továbbá nem képes érzelmek kimutatására. [4]

### 2.1.2 Nyelvi modellek fajtái

A nyelvi modellek tanulmányozása során számos felhasználási területet fedeztem fel. Van köztük szórakoztató csevegést biztosító, informatív. Ezen kívül bizonyos munkakörökben asszisztálhat, sok modell van specifikálva online rendelésekre (szállás foglalás, étel rendelés), melyek segítségével 7/24 éjjel-nappali támogatást nyújtva számos vállalkozás számára. Emellett adatkezelésben, adat elemzésben, személyre szabott reklámkategóriák kiválasztásában és még orvosi területeken is segítséget tud nyújtani. Például a megadott tünetek alapján fel tud állítani diagnózisokat, melyeket jelenleg egy orvosnak jóvá kell hagynia, mivel a jogi lépések a felelősség terhének témakörében ezen területek fejlődésével nem tud megfelelően lépést tartani. [5]

### 2.1.3 A chatbot megvalósítási típusai

Hangalapú botok, hibrid chatbotok, közösségi üzenetküldő chatbotok, menü alapú chatbotok, készség alapú chatbotok, kulcsszó alapú chatbotok, AI alapú chatbotok

#### 2.1.3.1 AI alapú chatbotok

Az AI-alapú chatbotok képesek felismerni a csevegés kontextusát és megértik a felhasználói kérdés valódi jelentését. Visszaidézik a korábbi interakciókat és használják azokat az információkat, hogy a kommunikáció során jelentésüket megőrizzék, különösen visszatérő ügyfelekkel való beszélgetések esetén. A kontextusfüggő chatbotok biztosítják, hogy az ismétlődő felhasználók folyamatosan

konzisztens élményt kapjanak. Továbbá, tárolhatják és felhasználhatják a különböző platformokon és csatornákon gyűjtött információkat a felhasználói szándékokkal kapcsolatban, hogy minden érintkezési ponton a fogyasztói igényeknek megfelelő kontextusban történjen a beszélgetés.

Ezen típusú chatbotok kapcsolódnak egy webhelyhez vagy alkalmazáshoz központi adatbázis révén, gyakran ügyfélkapcsolat-kezelő (CRM) rendszerhez vagy ügyféladat-platformhoz (CDP). Ez lehetővé teszi számukra, hogy fontos információkat szerezzenek a csevegésben részt vevő személyről, például a személy nevét, tartózkodási helyét vagy korábbi vásárlási előzményeit.

#### 2.1.3.2 Hangalapú botok

Egy hangalapú bot egy hangból szöveget és szövegből beszédet alkotó kommunikációs csatorna. AI és natural language understanding (NLU) hajtja ezeket a modelleket. Ez a technológia felismeri a beszéd kulcs szavait és ez alapján dönti el a megfelelő választ rá.

#### 2.1.3.3 Hibrid chatbotok

Egy hibrid chatbot kombinálja az élő csevegés és a chatbotok legjobb tulajdonságait. Egy ilyen modell lehetővé teszi a folyamatos ügyfélszolgálatot. Nagyban megkönnyíti ezen munkát végző emberek dolgait, mivel a beérkező kérdések elsősorban a chatbothoz érkeznek és az általános problémákat, kérdéseket sikerrel tudják kezelni. Ennek a megoldásnak az az erőssége, hogy ha a chatbot nem képes megoldást találni az ügyfél problémájára vagy a kérdés bonyolultsága meghaladja a tudását, akkor átirányítja egy élő munkatárshoz, aki segítséget tud nyújtani ebben az esetben.

#### 2.1.3.4 Közösségi üzenetküldő chatbotok

Az új közösségimédia-felületek térnyerésével ma már a vállalatok AI algoritmust is alkalmazhatnak ügyfeleik által preferált üzenetküldési platformokon. Ide tartozik például a Facebook Messenger, a Twitter és az Instagram, valamint az olyan üzenetküldő alkalmazások, mint a WhatsApp és a WeChat. Ez könnyedebb és kellemesebb online élményt tesz lehetővé az ügyfelek számára. Ezen kívül nagyobb elkötelezettséget a vállalat részére, mindezt anélkül, hogy növelné a kapcsolattartó központok munkáját.

#### 2.1.3.5 Menü alapú chatbotok

A legkezdetlegesebb chatbot típus a menü vezérelt navigáción alapul. Ezek a chatbotok legtöbbször egy rögzített döntési fát követnek, amely kattintható gombok formájában jelenik meg a fogyasztó számára. Ezek a chatbotok arra kérik a felhasználót, hogy válasszon többször és kattintson a megfelelő lehetőségekre. Ezzel eljutva a végső megoldáshoz. Fontos megjegyezni, hogy a menüalapú chatbotok a leglassabban biztosítanak valódi értéket a fogyasztónak, de egyszerűek és megfizethetőek az induláshoz.

#### 2.1.3.6 Készség alapú chatbot

Ez egy másik fajta bot, amely egy adott feladatsort képes végrehajtani, ha már előre meghatározott készségfejlesztő szoftverrel lett bővítve. Például a chatbot képes lehet időjárási információkat szolgáltatni, lekapcsolni a szoba világítását, ha okos háztartási készülékhez csatlakozik, online élelmiszereket rendelni stb. A készség bot forráskódjához való hozzáféréssel a fejlesztők saját készségfejlesztő chatbotokat készíthetnek, és integrálhatják azokat más platformokra.

#### 2.1.3.7 Kulcsszó alapú chatbotok

Ezek a chatbotok testre szabható kulcsszavakat és NLP-t használnak a beszélgetés során fellépő cselekvésre felszólító parancsok észlelésére, hogy megértsék, hogyan kell megfelelően reagálni a fogyasztók üzeneteire. Az NLP fogalma a későbbiekben kifejtésre kerül.

Egyre népszerűbbek a kulcsszó-azonosítást és a menü- vagy gombalapú navigációt kombináló chatbotok. Ha a kulcsszóészlelési funkció meghibásodik, vagy a felhasználónak további segítségre van szüksége a válasz megtalálásához, az ilyen chatbotok lehetőséget adnak a felhasználóknak, hogy a kattintható navigációs gombokon keresztül közvetlenül adjanak meg parancsokat. Ez egy hatékony megoldás, ha a bot nem tudja felismerni a kulcsszavakat a beírt bevitelben.

Ezeknek a modelleknek két jelentősebb típusa van, amelyek nagyban eltérnek mind felhasználásban mind megvalósításban. Ez a két típus a szabály- és az AI alapú chatbotok.

#### 2.1.3.8 Szabály alapú chatbotok

A szabályalapú chatbot kiváló megoldás azoknak a cégeknek, akik előre tudják, hogy milyen típusú kérdésekkel fognak találkozni ügyfeleiktől. A chatbot működése

alapvetően az if/then logika felhasználásán alapszik és előre meghatározott nyelvi követelményeket kell beállítani. A szavak, szószerkezetek, szinonimák és más hasonló elemek kiértékelése az alapvető működési elvek alapján történik. Az ügyfelek gyors és pontos segítséget kapnak, amennyiben az érkező kérdés a meghatározott paraméterek közé tartozik. [6]

## 2.2 Természetes Nyelvi Feldolgozás (NLP)

A Természetes Nyelvi Feldolgozás (NLP) egy olyan számítógépes megközelítést alkalmazó terület, amely egyaránt épül elméletekre és technológiákra a szövegek elemzéséhez. Mivel aktív kutatási és fejlesztési terület, nincs egyetlen elfogadott definíció, amely mindenki számára kielégítő lenne. Azonban vannak olyan szempontok, amelyek minden szakértő által elfogadottak. [7]

### 2.2.1 NLP definíció

A Természetes Nyelvi Feldolgozás elméletileg motivált terület, amely számítási technikákat alkalmaz a természetes nyelvű szövegek elemzésére és reprezentálására. Célja az emberi nyelvi feldolgozás emberszerűségének elérése, több szintű nyelvi elemzéssel, és számos feladathoz vagy alkalmazáshoz használható. Az AI alapú chatbotok pontosan NLP-t használnak. [8]

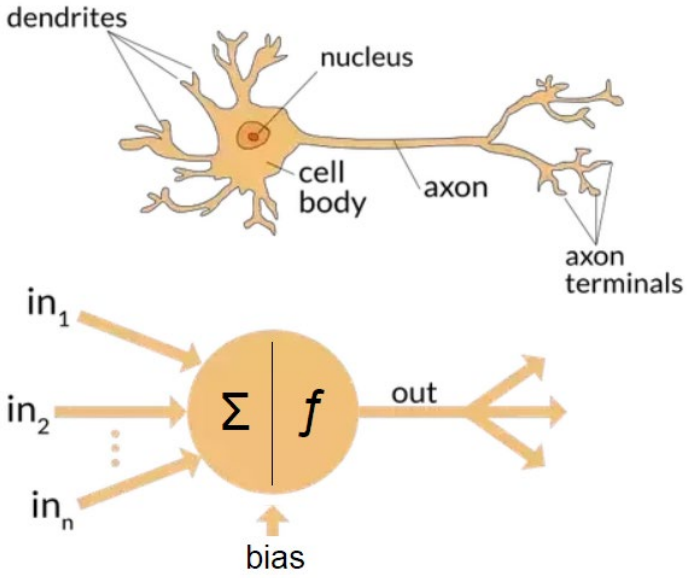
### 2.2.2 NLP könyvtárak

Több NLP könyvtár is létezik, melyek közül mindnek megvan az előnye és hátránya is. A legnépszerűbb NLP könyvtárak a Natural Language Toolkit (*NLTK), Gensim, CoreNLP, spaCy, TextBlob, Pattern* és a *PyNLPI*. Kiemelnék ezek közül kettőt, mivel ezen könyvtárak nagy része általános NLP feladatokat képesek ellátni. Az *NLTK* azért kiemelendő, mivel talán ez a legnagyobb felhasználóbázissal és tartalommal rendelkező mind közül. A másik, amit kiemelnék, az a *spaCy*, ami amellett, hogy hasonlóan hasznos funkciókkal rendelkezik, mint az imént említett könyvtár, kiemelkedő mennyiségű szöveget képes megérteni. Emellett tökéletes segítség a mély tanulás implementálásához, mivel több mint 49 nyelvet tud feldolgozni és ezt mind figyelemre méltó sebességgel teszi. [9]

## 2.3 Mesterséges Neurális Háló

A mesterséges neurális hálózat (*ANN*) egy olyan matematikai modell, amely az agyban található idegsejtek (neuronok) működését imitálja. Ez egy olyan rendszer, amely több egységből (neuronokból) és ezek közötti kapcsolatokból áll. Az

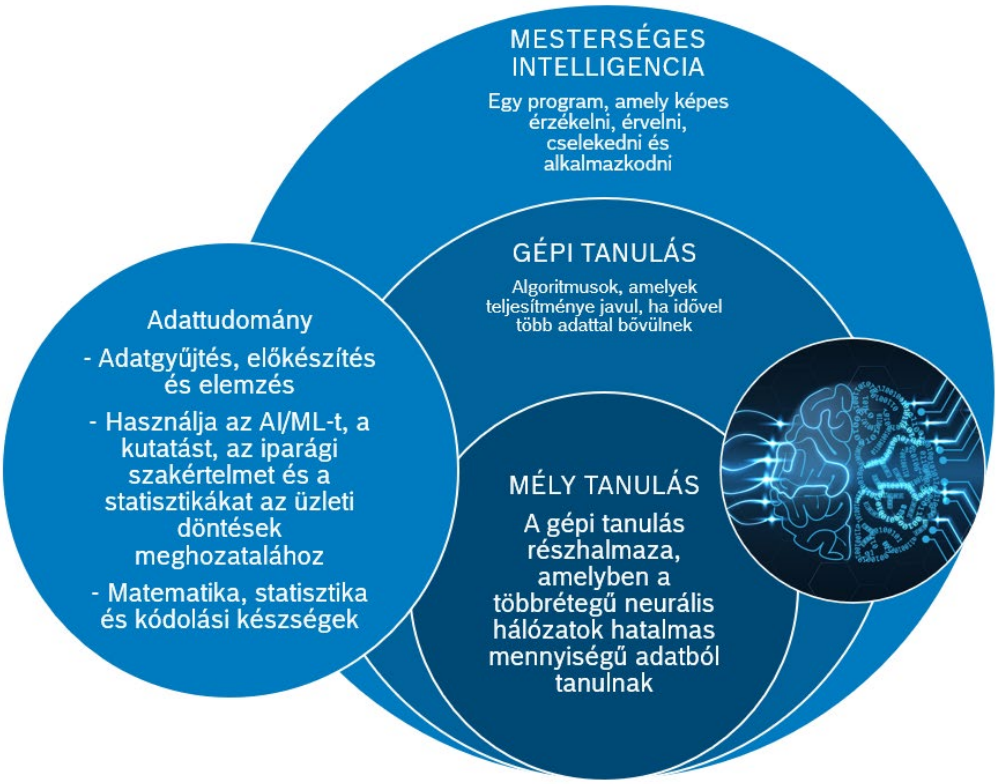
egységek információkat vesznek fel, feldolgozzák azokat és továbbítják a következő egységeknek, a neuronokhoz hasonlóan.




*3. ábra: Neuron és mesterséges neuron ábrázolva [10]*

## 2.4 Mesterséges intelligencia

A mesterséges Intelligencia (*MI*) angol nevén Artificial Intelligence (*AI*), olyan program, amely arra lett programozva, hogy reagáljon különböző dolgokra, az adatok alapján képes legyen döntéseket hozni, cselekedni. Ezekkel a tulajdonságokkal próbálják azt imitálni, ahogy az ember képes intelligens és ésszerű döntéseket hozni. A mesterséges intelligencia egy nagy és tág fogalom, melynek több részhalmaza is van. Ezek közé tartozik a gépi tanulás és ennek a részhalmaza a mély tanulás.




*4. ábra: Mesterséges intelligencia és annak részhalmazai*

### 2.4.1 Gépi tanulás

A gépi tanulás a mesterséges intelligencia részhalmaza. Lehetővé teszi a számítógépeknek, hogy tanuljanak és döntéseket hozzanak adatok alapján anélkül, hogy erre kifejezetten programoznák őket. Az algoritmusok képesek felfedezni adatokban rejlő mintázatokat és ezek alapján feladatokat megoldani, előrejelzéseket adni és új információkat generálni. A gépi tanulás számos valós életszerű alkalmazási lehetőséget kínál. Kulcsszerepet játszik az adatelemzésben, az automatizált döntéshozatalban, továbbá a képfelismerésben és a természetes nyelvfeldolgozásban is. [11]

### 2.4.2 Mély tanulás

A mély tanulás a gépi tanulás egy részhalmaza, amely lényegében egy három vagy több rétegből álló neurális hálózat. Ezek a neurális hálózatok arra törekednek, hogy utánozzák az emberi agy viselkedését, bár távolról sem érik el annak teljes képességeit. A mély tanulás lehetővé teszi a gépek számára, hogy "tanuljanak" nagy mennyiségű adatból. Míg egy egyszerű egyrétegű neurális hálózat csak közelítő előrejelzéseket képes készíteni, a további rejtett rétegek segíthetnek az optimalizálásban és a pontosság finomításában. A mély tanulás és a rejtett réteg

elnevezés onnan ered, hogy egy bonyolultabb és nagyobb mesterséges neurális hálónál nyomon követés nélkül, úgymond „rejtve” folyik ezekben a rétegekben a számítás. A mély tanulás elnevezés is ezt tükrözi, mivel ahogy egy nagyon mély verembe se látunk bele, itt sem követjük nyomon az egész folyamatot. [12]

## 2.5 Flask and Django

A két leggyakoribb webfejlesztéshez használt keretrendszer python programozási nyelvhez a *Flask* és a *Django*. A *Django* folyamatosan támogatott könyvtárak nagy mennyiségű csoportjának köszönhetően, egyszerű a használata, viszont ezt a keretrendszert nagyobb alkalmazások fejlesztéséhez alkalmazzák. Ezért az én jelenlegi elképzeléseimhez nem lenne optimális, a másik keretrendszerrel ellentétben. A *Flask* remekül illik a projektem megvalósításához egyszerűsége, rugalmassága, testreszabhatósága, valamint gyors és könnyű megérthetősége okán. [13] [14]

## 2.6 Kutatás eredménye

A Bosch multinacionális környezete miatt minden alkalmazott máshogyan közelít meg egy adott feladatot, így mindenki máshogy fogalmazza meg a kérdését a témával kapcsolatban. Pontosan ezért van szükség egy AI alapú chatbotra. Mivel a feladatot adatbiztonsági okok miatt nem tudjuk megvalósítani a *chatGPT* modelljével, ebből adódóan szükséges egy saját chatbot megalkotása. A mesterséges neurális háló megalkotásában nagy segítségemre volt a „Neural Network from Scratch in Python” című könyv Harrison Kinsley és Daniel Kukiela-tól, amit részletesen tanulmányozva, sikeresen meg tudtam alkotni a saját neurális hálómat.

# 3 Feladat ismertetése

A probléma alapja: vannak alkalmazottak a cégnél, akik ritkán vagy még soha nem végeztek el egy adott általános jellegű feladatot. Ezeknek a megoldására rengeteg plusz kapacitás szükséges. Nem ritkán több órás kutatás szükséges egy feladat elvégzéséhez a nagyvállalati környezet miatt.

Ennek a megoldására szeretnék elkészíteni egy chatbotot, amely a korábban feltett kérdésekre tudni fog választ adni, ezzel megkerülve a bonyolult és időigényes utánajárást minden alkalommal, mikor valaki felteszi a már korábban felvetült kérdéseket. Az AI alapú chatbot nyelvi modelljének betanítását egy mesterséges neurális háló használatával oldom meg.

## 3.1 Programspecifikáció

Itt bemutatásra kerülnek a különböző adatok, képernyőtervek, adatszerkezetek és maga a szoftverkörnyezete is.

### 3.1.1 Adatok

Olyan adatok, melyeket a program kezelni fog vagy feladatokat végez el. Ilyen adatok a bemenő, törzs- és kimenő adatok.

#### *3.1.1.1 Bemenő adatok*

A bemenő adatokat, melyeket a chatbot fogad, a felhasználók, jelen esetben a cég dolgozói visznek be szöveges formátumban. Elsősorban kérdéseket vagy egy adott témával kapcsolatos érdeklődést vár inputként. Ezeket az adatokat a program előkészíti és átalakítja bináris formára, melynek jelentőségére később fogok kitérni.

#### *3.1.1.2 Törzsadatok*

A chatbot által használt törzsadat egy előre megadott információhalmaz, témák szerint kategorizálva a céggel kapcsolatosan, egy *json* típusú szöveges fájlban tárolva. Ezek az adatok, például hasznos linkek cégen belüli weboldalakra, utasítások a vállalaton belüli folyamatok elvégzéséhez és általános információk szerteágazó témakörökben. Mivel ezek az adatok ritkán változnak, ezért is hasznos az, hogy külön fájlban vannak tárolva és ezekből olvassa be a chatbot a megfelelő választ az bemenő adatra.

#### *3.1.1.3 Kimenő adatok:*

Ezek egy külön *json* fájlba vannak lementve, melyből a választ szövegként írja ki a felhasználónak. Kimenő adatként tulajdonképpen egy valószínűségi mátrixot kapunk. Ebben a mátrixban minden sor egy vektor. A vektorok a válaszként kapható témakörök számával egyeznek meg. Ezen témakörök valószínűségei vannak reprezentálva, vagyis ezeknek az összege mindig 1-et tesz ki és az adott indexel rendelkező valószínűség azt adja meg, mennyi az esélye az adott kimenetnek. Például, ha van öt témaköröm, akkor a kimenet egy lehetséges értéke így néz ki: [0.2 0.1 0.2 0.4 0.1]. Ennek a példának a kimeneti értéke lesz a válasz az adott kérdésre, amely itt a hármas indexű témakör válaszai közül választ ki egyet (abban az esetben, ha több megfogalmazása is van a válasznak).

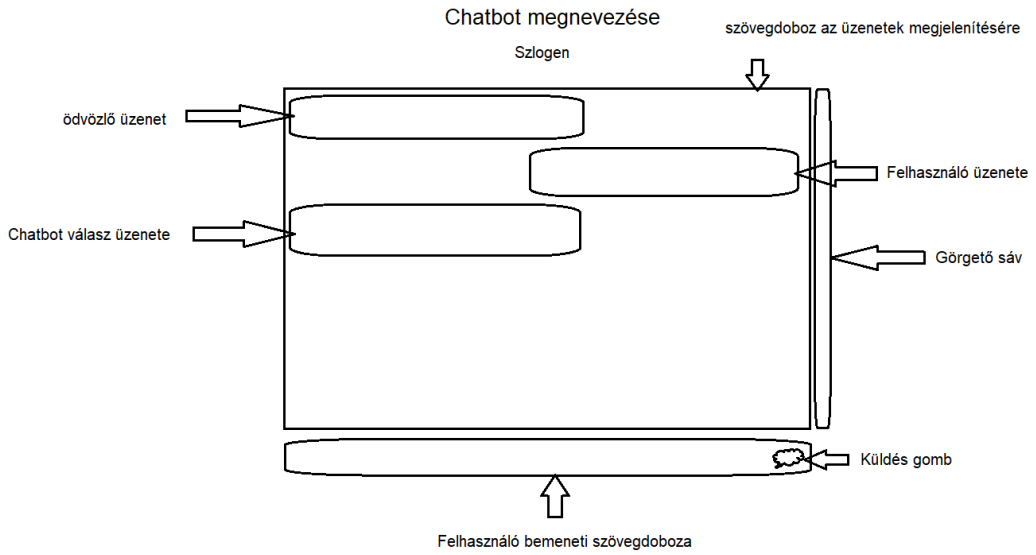

*5. ábra: Chatbot üzenőfelülete*

#### *3.1.1.4 Példa a chatbot be- és kimenő adataira*

A fenti képen látható, hogy a felhasználó elküldi a „hello” bemeneti üzenetet, melyet a chatbot az előzetesen betanított neurális hálón átenged és a számításokat elvégezve, kiadja a témakörökre számolt valószínűséget. Láthatóan a betanítás megfelelő volt ahhoz, hogy felismerve az üzenetet a „hello” inputra a „greeting” témakört tartja a legesélyesebbnek, ezért az ahhoz előre megírt válaszok közül a „Hi there!” üzenetet adta válaszul.

### 3.1.2 Képernyőtervek

A chatbot egy letisztult és a lényegre fókuszáló felhasználói felületet UI-t kapott.

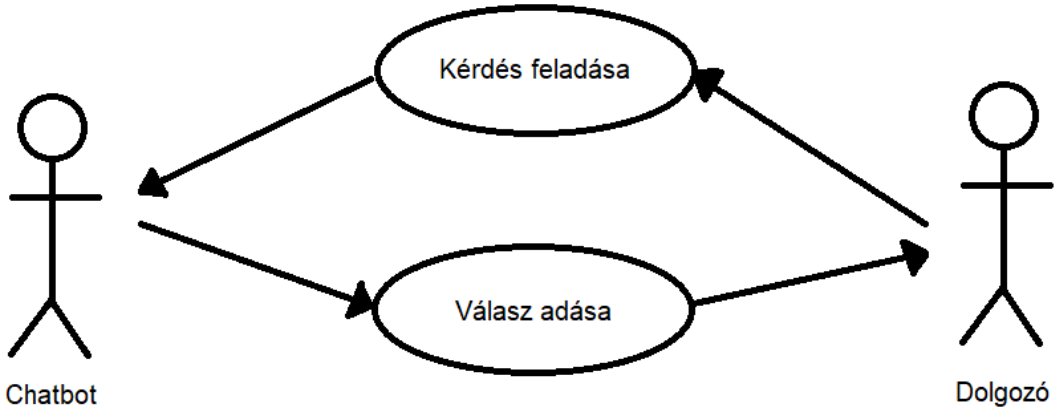



*6. ábra: Chatbot alkalmazás képernyőterve*

A demó alkalmazás nem fog mást tartalmazni, mint a nevét, egy szlogent, üzenőfelületet és egy bemeneti mezőt, melyhez egy küldésgomb is tartozik. A színek és a tartalom jelenleg demó változatban van. A jelenlegi verzióban hozzám közelálló színeket és árnyalatokat használtam, Arial betűtípussal és alapértelmezett betűméretekkel.

### 3.1.3 Adatszerkezet

Egy használati eset diagramon szemléltetve a rendszer az alábbi módon néz ki:




*7. ábra: Használati eset diagram*

A Dolgozó bemenetként megad egy kérdést a Chatbot ezt feldolgozza és Választ ad rá, amelyet kiiratunk a felhasználónak. Egy gyorsbillentyű funkcióval lehetővé teszem az üzenet elküldését az „Enter” gomb lenyomásával is.

### 3.1.4 Használati mód

A felhasználói felület egyenlőre semmi mást nem tartalmaz, csak egy üzenet dobozt a beszélgetés megjelenítésére, egy bemeneti mezőt a felhasználó kérdéseinek felvételére és egy küldésgombot, amely a feltett kérdést továbbítja a chatbotnak.

### 3.1.5 Szoftver környezet

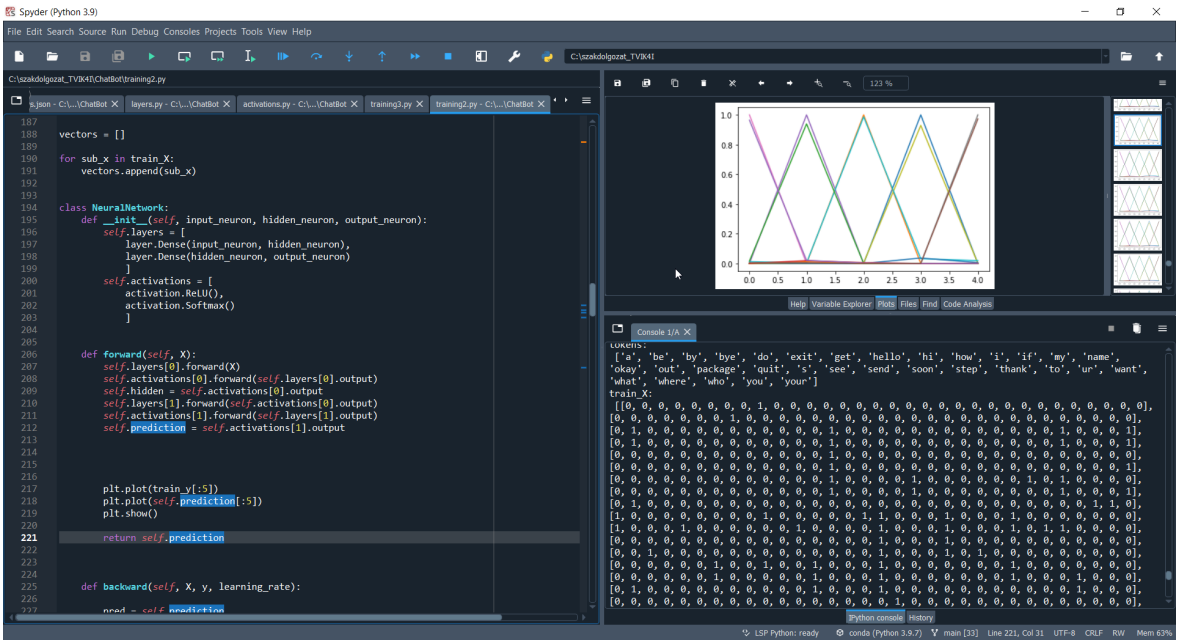
Az alkalmazáshoz használt szoftverek mind ingyenesen elérhetőek és nyílt forráskódúak. A Chatbotot minden böngészőből el lehet majd érni teljesen ingyenesen egy link segítségével.

## 3.2 Fejlesztő eszközök

Fejlesztői környezetnek elsősorban a *Spyder IDE*-t használtam, mivel ezt ismertem meg a cégnél, mint fejlesztő eszköz és első látásra magával ragadott az átláthatósága és a rengeteg hasznos implementációja, melyek nagyban megkönnyítik a fejlesztés folyamatát. Ebben a környezetben a *spaCy NLP* nyelvfeldolgozó csomagjait használom fel segítségként. A webalkalmazáshoz szükséges *HTML* (HyperText Markup Language) és a stílus formázáshoz elengedhetetlen *CSS* (Cascading Style Sheets) kódokat, továbbá a backendhez a *Flask* keretrendszert a *VS Code*-ban fejlesztve készítem el.

### 3.2.1 Spyder

A *Spyder* egy tudományos *Python* fejlesztői közeg.Erőteljes interaktív fejlesztői környezet a *Python* programozási nyelvhez, amely fejlett szerkesztési lehetőségeket, interaktív tesztelési funkciókat, hibakeresési eszközöket és öndiagnosztikai funkciókat kínál. Numerikus számítási környezet az *IPython*-nal (fejlett interaktív Python interpreter) és a népszerű *Python* könyvtárakkal, mint például a *NumPy* (lineáris algebra), a *SciPy* (jel- és képfeldolgozás) vagy a *matplotlib* (interaktív 2D/3D ábrázolás) való támogatásnak köszönhetően. [15]




*8. ábra: Spyder fejlesztői környezet*

### 3.2.2 Python

A Python egy magas szintű, objektum orientált és strukturált programozási nyelv. Egyik jelentős tulajdonsága a dinamikus névfeloldás, ami azt jelenti, hogy a metódusok és a változók nevei a program futása közben kerülnek összekapcsolásra, a kód írásakor még nem. Az egyik leg könnyebben megérthető szintaxisú nyelvek közé sorolják, könnyen érthető, ezáltal gyorsan el lehet sajátítani a használatát. [16]

### 3.2.3 JavaScript

A *javascript* segítségével interaktív funkciókat adhatunk a weboldalakhoz. Ezzel a programozási nyelvvel írt kód a kliensoldalon fut, ezáltal interaktív és dinamikus elemekkel, animációkkal tudja kibővíteni a weboldalunkat. Dokumentum Objektum Modell (*DOM*) manipulációt tesz lehetővé manipulálva az oldal HTML és CSS elemeit, emellett a szerverrel való kommunikációt is. Tehát a felhasználói interakciókra reagálva dinamikus tartalomváltozást tesz lehetővé. [17]

## 3.3 A felhasznált csomagok bemutatása

Az alábbi pontokban bemutatom a felhasznált csomagokat részletekbe menően. Ezek segítségemre voltak a program sikeres és gyors megvalósításában. Szó lesz a *slaCy*, a *NumPy*, a *matplotlib* és a *random* csomagról is.

### 3.3.1 spaCy

A *spaCy* egy nyílt forráskódú *Python* könyvtár, amely lehetővé teszi a természetes nyelvfeldolgozást (NLP*)* egy nagy méretű szövegen, meglehetősen gyors sebességgel. [18]

### 3.3.2 NumPy

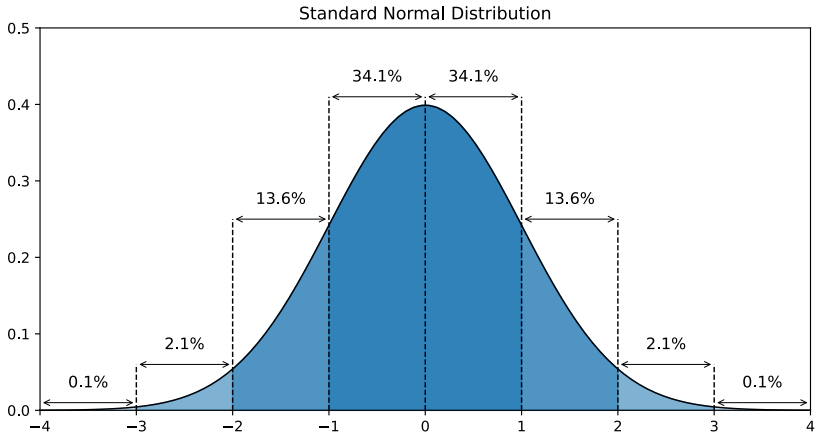
A *NumPy* lehetővé teszi az adatok hatékony manipulációját, elérését és kezelését vektorokon, mátrixokon és többdimenziós tömbökön is. Továbbá a matematikai műveletek végrehajtását, a statisztikai elemzéseket és a nagy adatkészletek kezelését, ami elengedhetetlenül fontos a tudományos kutatásban és adatelemzésben. A *NumPy* a *Python* nyelv elsődleges tömbprogramozási könyvtára, és rendkívül fontos szerepet játszik a kutatási és elemzési folyamatokban számos területen, mint például a fizika, kémia, csillagászat, földtudomány, biológia, pszichológia, anyagtudomány, mérnöki tudomány, pénzügy és közgazdaságtan. [19]

### 3.3.3 Matplotlib

A *Matplotlib* egy *Python* könyvtár 2 dimenziós gráfok ábrázoláshoz. Támogatja az interaktív és a nem interaktív ábrázolást is, továbbá képes a képeket elmenteni többféle kiterjesztési formában is (PNG, PS és mások). [20]

### 3.3.4 Random

A *random* csomagban felhasználásra kerülő funkció a *randn*, amely random generál normál eloszlás alapján számokat. Az alábbi ábrán látható egy normál eloszlás ábrázolása.




*9. ábra: Standard normál eloszlás ábrázolása [21]*

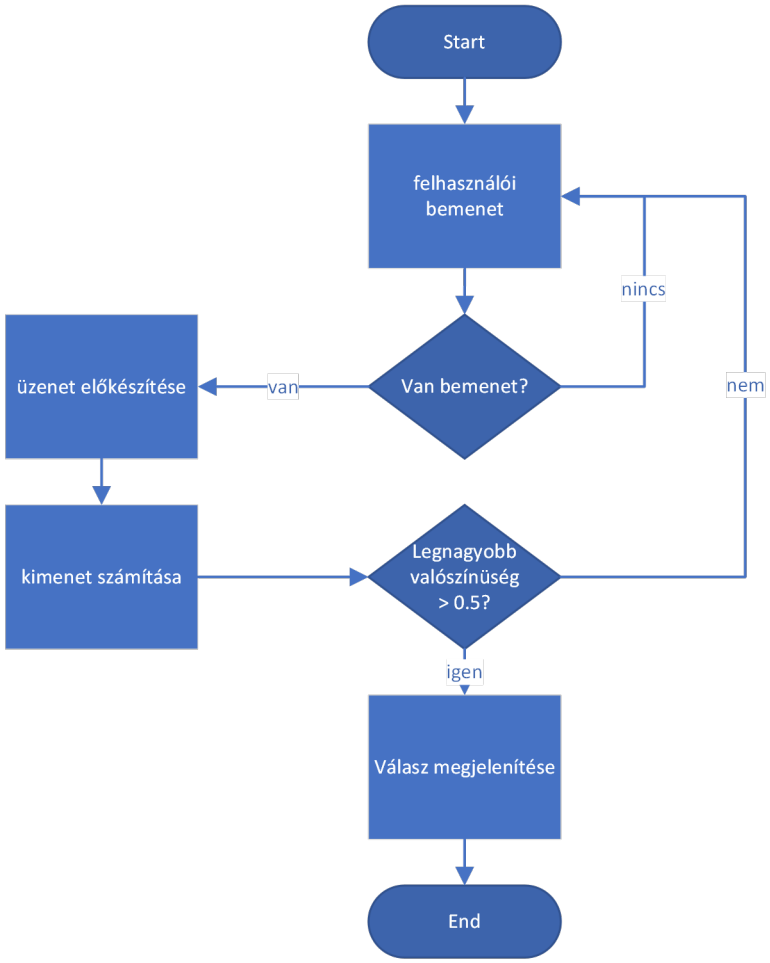
# 4 Megvalósítás

A következőkben ismertetem a terv részleteit, a felhasznált fejlesztői eszközök elérhetőségét, telepítését és magát a program kódolását is részletesen bemutatom.

## 4.1 A terv ismertetése:

A korábban említett összehasonlításokban elemzett eszközök közül a *spaCy NLP* könyvtáraival *Spyder* és *VS Code IDE*-vel, továbbá *Flask* keretrendszerrel fogom megvalósítani a chatbotomat webalkalmazásba implementálva. A program egy weboldalra fog épülni, tehát állni fog egy *HTML* fájlból és egy *CSS* stílusfájlból is. Tartalmazni fog egy *Flask* alkalmazást, amivel kezelem a kérés és válasz folyamatokat. Egy training és egy a chatbotot megvalósító fájl, továbbá az ezekhez szükséges intentek, melyek a betanításhoz szükséges adatokat tartalmazzák *json* kiterjesztésben. A rétegek és az aktivációs függvények megvalósításait külön-külön fájlban valósítom meg.

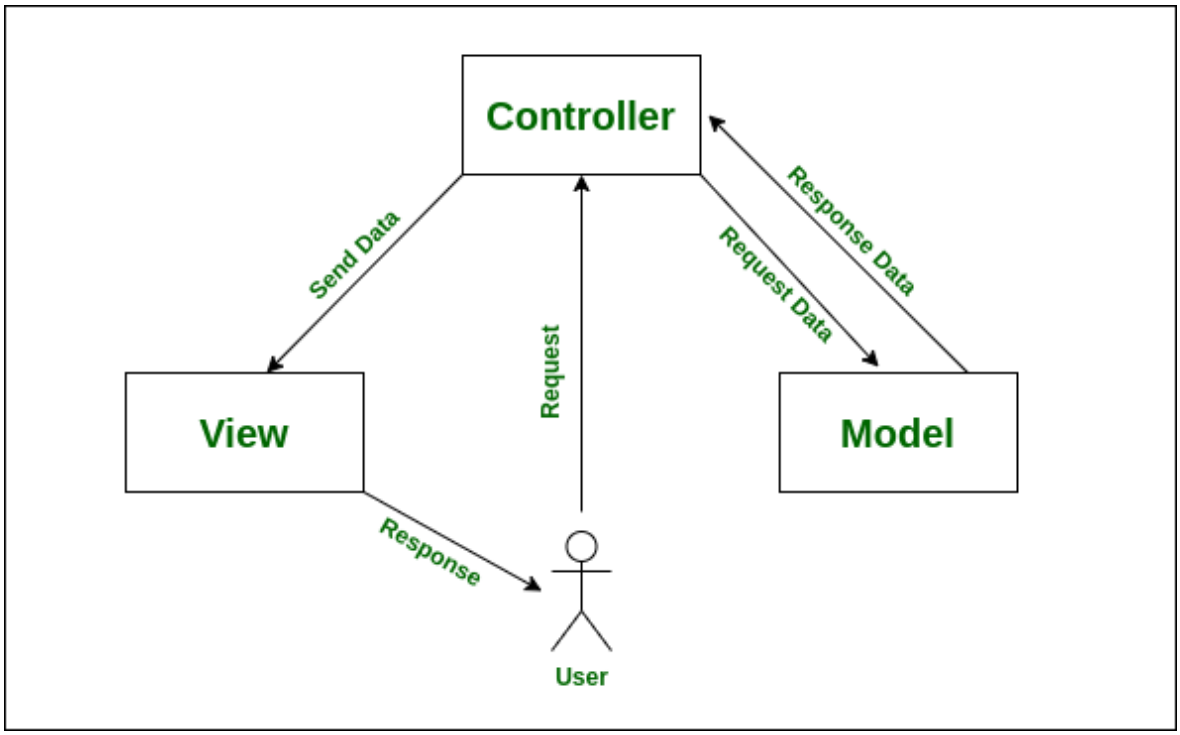
A *spaCy* angol nyelvi modelljének segítségével fogom kivitelezni a chatbotom betanítását, mivel a Bosch egy multinacionális vállalat. Az RBHM körein belül is 10%-os a Magyarul nem beszélő kollégák száma, mivel egy minden alkalmazott számára elérhető szolgáltatást szeretnék nyújtani, ezért választottam az angol nyelvet.




*10. ábra: A program folyamatábrája*

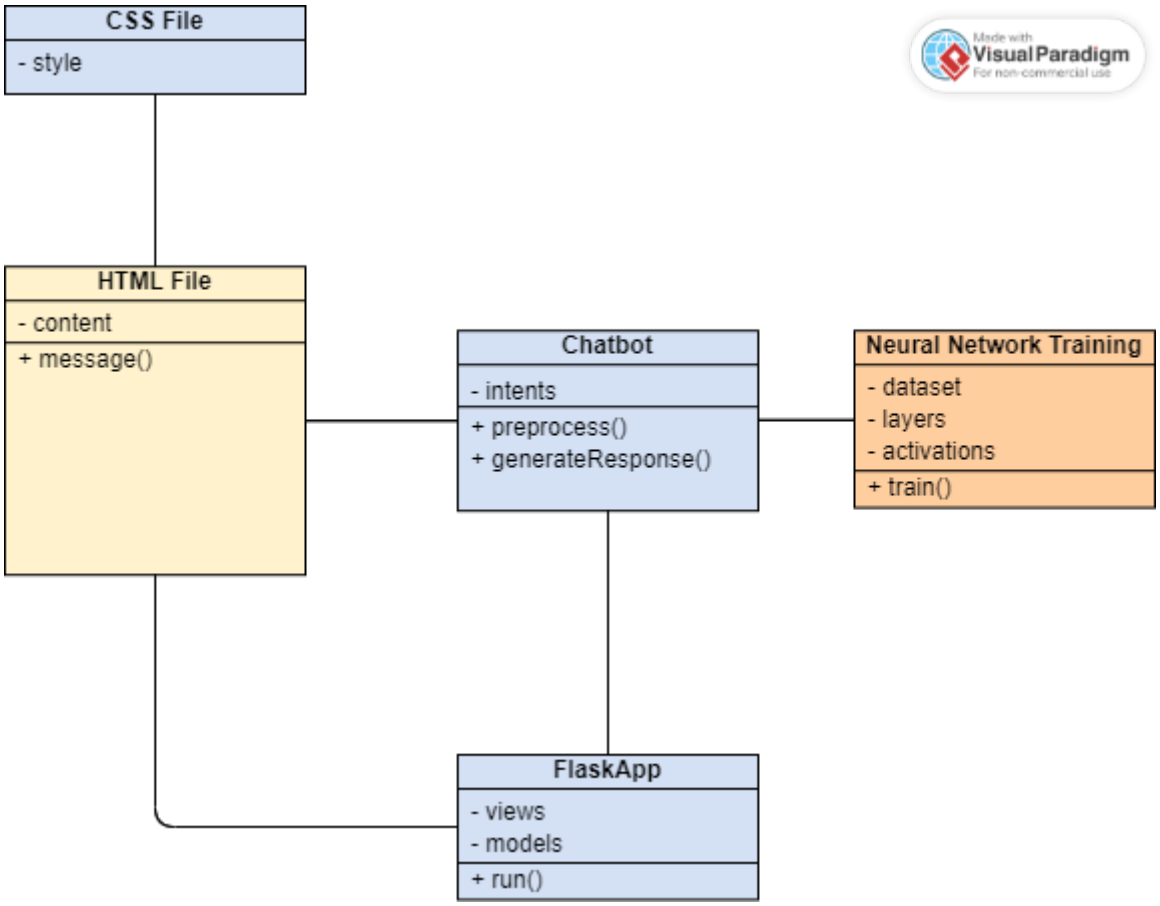
A *HTML* kódot *Javascript* fogja összekapcsolni a *Flask* applikációval. A *CSS* a *HTML* kódban meghívva fogja formázni, stílust adni az oldalnak. Az intents, layers és activations fájlok segítségével fogom a betanítást végezni a training.py fájllal. Három darab *json*-ben lesznek lementve a kapott adatok: egy model, ami a kiszámított súlyokat és eltolásokat tartalmazza, egy words, melyben a chatbot szókincsét mentem el és egy labels, amiben a témakörök vannak elmentve.

Ezeket követően, ha a felhasználó küld egy üzenetet, azt a *Flask* továbbítja a chatbotnak egy GET kéréssel (request) és a felületen megjeleníti a feltett kérdést. A modell paramétereit felhasználva megjósol egy választ, amit elküld a *Flask*-nak egy POST kéréssel (response) és ezt a *javascript* által létrehozott *HTML* elem segítségével dinamikusan kirajzoljuk a weboldalra.




*11. ábra: Model-View-Controller módszer mintája [22]*

A kommunikációhoz szükséges kapcsolatot a felhasználó és a chatbot között a már tanult Model-View-Controller struktúráját szemelőt tartva igyekeztem kivitelezni a rendszer felépítését.




*12. ábra: Chatbot szerkezetének osztálydiagramja*

## 4.2 Fejlesztőeszközök beszerzése

Először az *Anaconda* nevezetű nyílt forráskódú, felhasználóbarát, megbízható csomag és menedzsment rendszert telepítettem. A *Conda* elsősorban a programozás tudományos oldalát hivatott segíteni, többek között a gyors és egyszerű csomag telepítéssel, futtatással és frissítéssel, nem beszélve a külön létrehozható fejlesztői környezetek hasznosságáról. A fejlesztői környezetek létrehozásának nagy előnye, hogy kiküszöböli a verziókülönbségek okán felmerülő hibákat. Az *Anaconda* navigátorából elérhetővé válik sok hasznos fejlesztő program letöltése például a *JupyterNotebook* vagy az általam választott *Spyder* alkalmazás.

Másodszor a *VS Code* nevű kód szerkesztő alkalmazást töltöttem le a *Visual Studio* hivatalos oldaláról. A *VS Code* egy nyílt forráskódú platformfüggetlen program. Nagy előnyei a közel korlátlan testreszabhatóság és a figyelemreméltó számmal bíró bővítménykészlete, amely mögött egy hatalmas közösség és rajongótábor áll.

Az *Anaconda* promt-ból lehetőségünk nyílik *python* csomagok egyszerű és gyors telepítésére a „pip install csomagnév” paranccsal. Sok hasznos csomag áll

rendelkezésünkre, melyeket támogat a *Conda*, ilyen a *NumPy, matplotlib, spaCy, pandas, SciPy*. Ezek közül az alábbiakban néhányuk telepítését részletezem.

### 4.2.1 NumPy telepítése

A *NumPy* segítséget nyújt a többdimenziós tömbökkel való műveletek elvégzéséhez. Telepítése *Anaconda* prompt-ból a „pip install numpy” parancs indításával végezhető el. A *Spyderben* történő meghívásával aliaszt is adhatunk neki, ami egyezményesen, de nem kötelezően az „np” szokott lenni.

A meghívása: „import numpy as np”.

### 4.2.2 Matplotlib telepítése

A *matplotlib* egy átfogó csomag statikus, animált és interaktív vizualizációk készítéséhez. Ez a csomag a tömbök vizualizációjához kell, ezzel átláthatóbbá és értelmezhetőbbé téve azok értékeit. Telepítése ugyan azzal a módszerrel működik, mint a *numPy* esetében a „pip install matplotlib” paranccsal.

A meghívása: „import matplotlib.pyplot as plt”.

### 4.2.3 spaCy telepítése

A *spaCy* telepítésére a természetes nyelvfeldolgozás miatt lesz szükség. Nagy méretű adatbázis van mögötte szavakból, azok szótári és a lehető legtöbb ragozott és toldalékolt formájukból. Ez lehetővé tette a mondatok és az azokat alkotó szavak feldolgozását. Telepítése a „pip install spacy” paranccsal történik.

A meghívása: „import spacy”.

Ezek után a szükséges betanított csővezetéket, ami jelen esetben az angol lesz, a következőképpen lehet feltelepíteni: „python -m spacy download en_core_web_sm”. Ezzel hozzáférést nyerünk az angol szavak és azok legtöbb alakját tartalmazó adatlistához. További nyelvek letöltésének instrukciói megtalálhatók a *spaCy* weboldalán. [23]

### 4.2.4 Flask telepítése

A *Vs Code* telepítését követően, feltéve, hogy a *python* telepítve van a számítógépünkre, a következő lépésekkel tudjuk letölteni a *Flask* keretrendszert. A „python -m venv venv” paranccsal létrehozunk egy virtuális környezetet. Ezt

követően a „venv\scripts\activate” beírásával aktiválhatjuk a létrehozott környezetünket. A „pip install flask” parancsot követően az alábbi képen látható módon lehet létrehozni az alkalmazást.

```
app = Flask(__name__, static_folder='static')

@app.route("/")
def home():
    return render_template("web-app.html")
```

*13. ábra: Kép egy alap Flask alkalmazásról*

Ha az előbb említett lépések megvannak, akkor a „flask run” parancs a terminálból indítva elindítja az alkalmazást.

## 4.3 Program megvalósítása

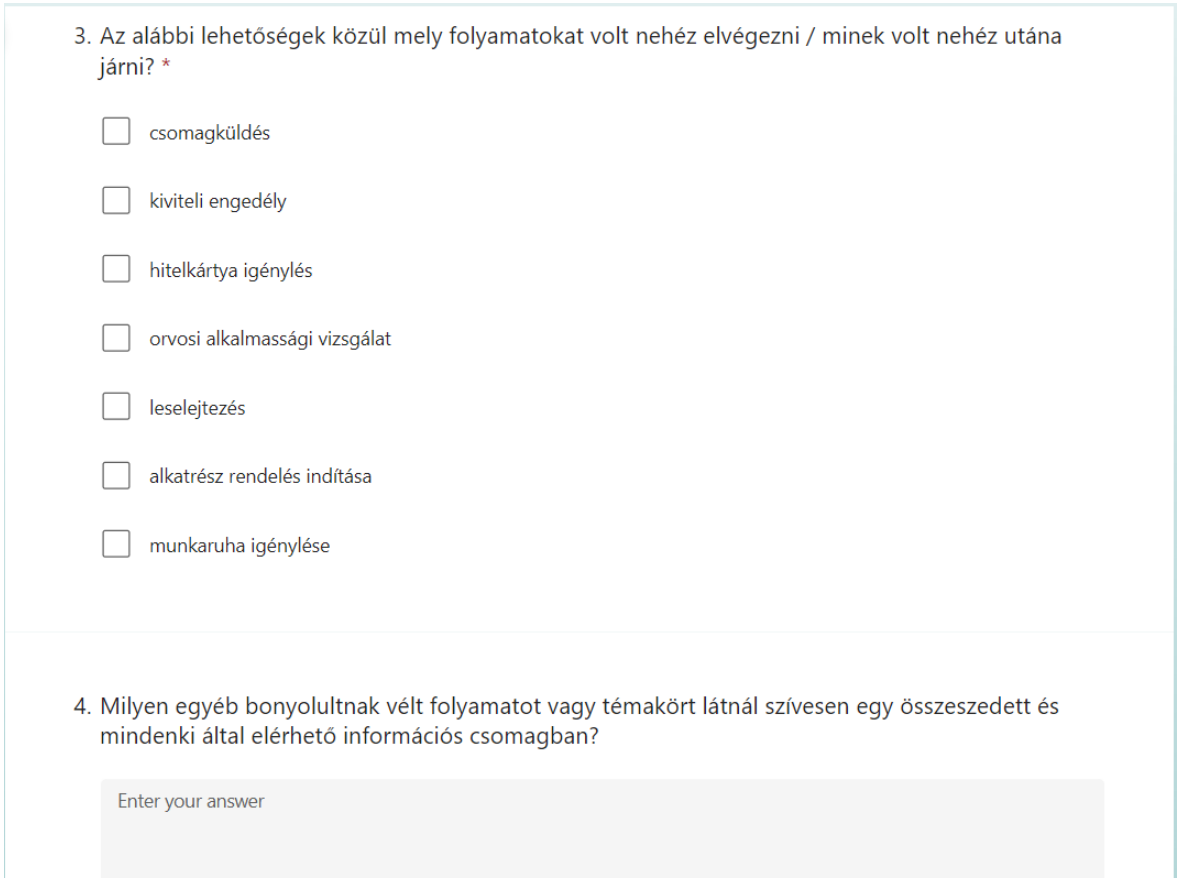
A megvalósítás sorrendjét a már tanult eldobható prototípus koncepciója alapján alakítottam ki.

Egy Chatbot alapvető és talán legfontosabb eleme a modell, melyet egy mesterséges neurális hálón tanítunk be. A megfelelő csomagok importálását követően a már korábban említett angol nyelvi modell betöltésével kezdem:

nlp = spacy.load("en_core_web_sm")

### 4.3.1 Adatgyűjtés

Az első feladat, maga a betanításhoz szükséges adatok összegyűjtése volt. Ennek összegyűjtésére felmérést végeztem az osztályunkon. A felmérés két részből állt melyekhez egy-egy office kérdőívet készítettem, amit az Outlook levelező rendszerén keresztül körbe küldtem mindenkinek. A kérdőívben felsoroltam pár általam fontosnak tartott kérdést és témakört, melyek felmerülhetnek egy újonnan érkező munkatárs fejében vagy bárkinek, ha olyan az adott feladat, melyet ritkán, vagy csak kevés ember végez el. Továbbá azt is ki kellett fejteniük, hogy ők, milyen további témaköröket tartanak kidolgozandónak, milyen kérdések fogalmazódnak meg bennük ezekkel a témakörökkel kapcsolatban.

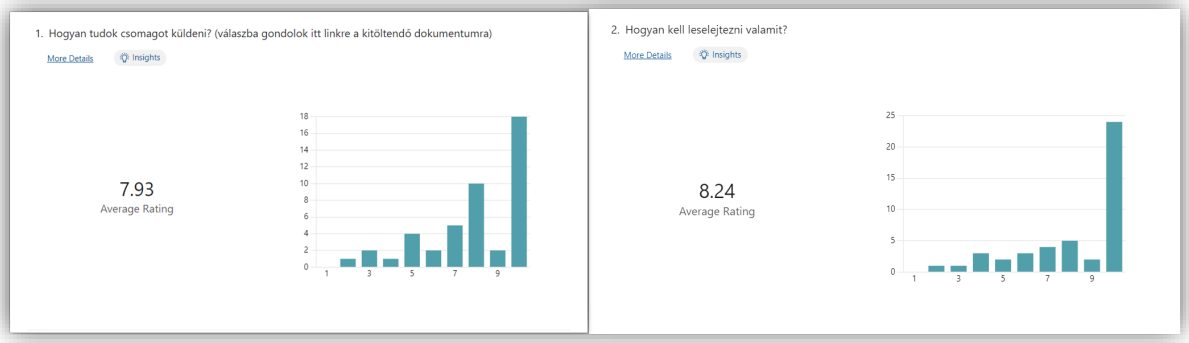



*14. ábra: Első körös kérdőív részlete*

Az ezekre kapott válaszok és információk alapján kidolgoztam egy több mint 20 témából és kérdésből álló listát, melyet kidolgoztam egy második körös kérdőívvé. Fontos kritérium volt a kérdőíveknél az is, hogy nem szabadott túl hosszúnak vagy nehezen kidolgozhatónak lenniük. Ezt az első körös kérdőívem hibájából tanulva, ahol két nagyobb kifejtendő kérdést is feltettem, a második kérdőívemben csak értékelni kellett az adott kérdést vagy témakört. Az elsőre kapott válaszok száma 24 fő volt, viszont a másodikra, az iménti hibámból tanulva már 45 választ kaptam. Ebből az a tanulság, hogy az emberek a lehető legkevesebb erő- és idő belefektetésével akarják megoldani a feladataikat. Jelen esetben egy kiküldött kérdőívre kapott válaszok száma attól függ, mennyire fontos a kitöltése az adott illetőnek, mennyire van ideje rá, mennyi időt vesz igénybe és mekkora a kitöltendő kérdések nehézsége.


*15. ábra: Második körös kérdőív részlete*

A végeredményben egy listát kaptam átlagolt pontokkal, arról, mennyire fontos az adott kérdés vagy témakör az osztály dolgozóinak.




*16. ábra: A kérdőív állapotai (részletek)*

Az összegyűjtött témakörök és azok fontossági sorrendbe állítása után a 10 legmagasabb pontszámot szerzőkre kidolgoztam a válaszokat, melyeket egy *json* fájlban mentettem el az alább látható formátumban.

```
{"intents": [
    {"tag": "greeting",
    "patterns": ["hi","hello"],
    "responses": ["Hi there!"]
    },
    {"tag": "name",
    "patterns": ["What is your name?", "What's your name?", "name", "Your name?", "whats ur name?", "whats your name?", "Who are you?"],
    "responses": ["I don't have a name yet...", "I'm just a ChatBot :("]
    },
    {"tag": "being",
    "patterns": ["What is you?", "What are you?", "you are?", "Are you a ChatBot?", "What kind of ChatBot are you?"],
    "responses": ["I'm a ChatBot.", "I'm a ChatBot assistant, I'm here to serve you useful informations about general questions of the company."]
    },
    {"tag": "travel",
    "patterns": ["Travel", "What to do before travelling?", "business trip"],
    "responses": ["""Contact the travel department or designated personnel responsible for managing travel arrangements.
                   Provide your travel details such as destination, duration, purpose, and any specific requirements.
                   Inquire about the necessary travel documents, such as passports, visas, or permits.
                   Follow any approval processes or guidelines set by the company for travel requests.
                   Obtain the required information regarding accommodation, transportation, and any travel allowances or reimbursements."""]
    },
    {"tag": "package_recieve",
    "patterns": ["How to get my package?", "Where to get my package?", "Where is my package?", "package"],
    "responses": ["If it's arrived, you can pick up it at the import section."]
    },
    {"tag": "workon",
    "patterns": ["How to start a workon?", "Workon start steps?", "workon"],
    "responses": ["""Contact the WorkOn administration or relevant department.
                   Express your interest in using WorkOn and provide details about your project or task.
                   Attend a WorkOn orientation or training session, if required.
                   Receive access credentials or user accounts for WorkOn.
                   Follow any guidelines or procedures provided by the administration for project management and collaboration using WorkOn."""]
    },
    {"tag": "deconmission",
    "patterns": ["How to get my package?", "Where to get my package?", "Where is my package?", "package"],
    "responses": ["""Determine the specific guidelines and policies for decommissioning items within your company.
                   Identify the item you wish to decommission and ensure it is no longer needed or functional.
                   Notify the appropriate department or personnel responsible for asset management or disposal.
                   Follow any specific procedures or documentation requirements for item decommissioning.
                   Coordinate with relevant parties to ensure proper disposal, recycling, or repurposing of the item, adhering to environmental reg
```

*17. ábra: Chatbot kidolgozott témakörök struktúrája*

Ebben a fájlban intenteknek nevezik azokat a témaköröket, melyeket a chatbot később meg tud különböztetni. A „tag” jelenti a témakör nevét, ezek tetszőleges megnevezések, viszont célszerű olyan nevet választani, amiről fel lehet ismerni a tartalmát. Ez teszteléskor jól fog jönni. A „pattern” elemnél azt adjuk meg, milyen kérdésekre, szavakra reagáljon a chatbot. Például a „hi” bemenetre, felismeri majd, hogy a „Hit here!” választ adja, mivel a többi „pattern” -ben nincs jelen a „hi” szó. A „response” elemek közül fogja a chatbot kiválasztani az egyik választ, miután eldöntötte, mi a legmegfelelőbb válasz az adott kérésre.

### 4.3.2 Adatok előfeldolgozása

A természetes nyelvfeldolgozás (NLP) módszerének segítségével hajtom végre az adatok előfeldolgozását.

A létrehozott intents.json-t megnyitom a json.load paranccsal és betöltöm egy python fájlba, ahol kulcs-érték párokként szerepelnek. Végig iterálok minden elemen és a szavakat egy listába szedem, ebből egy szótárat létrehozva. Eze,n a ponton van egy listám speciális *spaCy* elemekkel, mely tartalmazza a „pattern” összes szavát. Ezzel viszont még sok a teendő. A speciális elemeket úgy hívják, Token. Az írásjeleket kiszűröm a listából, aztán minden karaktert kis betűtípusra állítom át. Az utóbbira azért van szükség, hogy ne legyen különbség például az

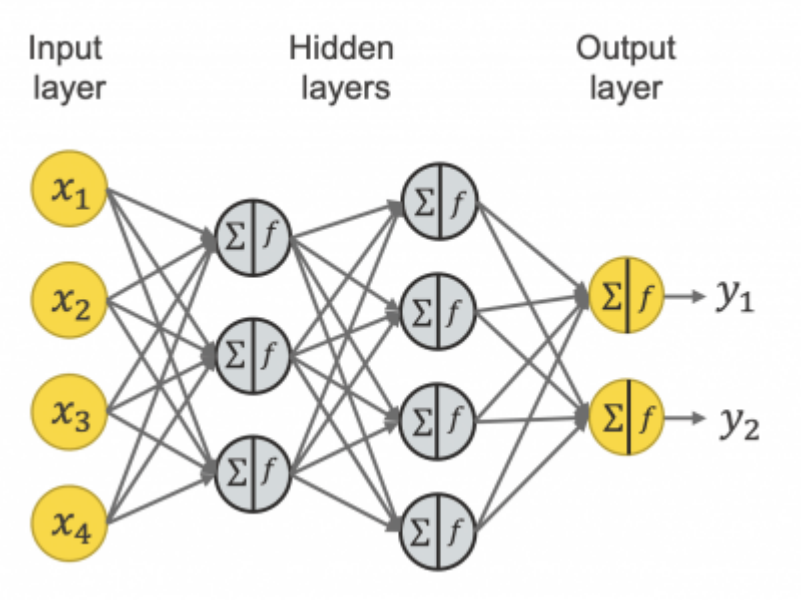
Alma, alma és az ALMA szó között, mivel tartalmát tekintve ugyan az mind a 3 szó, viszont a formai különbségek miatt 3 különböző szónak venné a program. Ezen kívül a duplikált szavakat kiszortírozom és sorrendbe rakom a listát. A kapott szavakat és címkéket *json* fájlokba mentem.

Kimenetnek létrehozok egy vektort és a címkék számával megegyező nullákkal töltöm meg. Az összes mondatot a „pattern” -ben felosztom szótári alakjukra és külön választom egymástól mindet. Következő lépésben végig iterálok a korábban létrehozott szótáram elemein és ha a „pattern” egyik szava megegyezik vele, akkor az adott indexű elem értékként 1-et kap, ha nem egyezik, 0-t.


*18. ábra: Mondatok bináris listává alakítva*

A fenti képen remekül látni, a példa szótárban, hogy mi a szótárunk tartalma és alatta a „pattern” mondatai. Minden mondathoz hozzárendelem, melyik címkéhez tartozik és a bináris kódját, amit az határoz meg, mely szavak vannak jelen a mondatunkban és a szótárunkban is egyaránt. Ezeket a listákat „tuple” -ként párosával a „training” listába szedem. A „train_x” lesz a példamondatok listája, a „train_y” pedig a hozzájuk tartozó tag-ek listája.

### 4.3.3 Mesterséges neurális háló megvalósítása




*19. ábra: Mesterséges neurális háló leképzése [24]*

A mesterséges neurális háló a betanítás legfontosabb része. Ebben az osztályban először is definiálunk egy „__init__” funkciót, amely megegyezik a *Java*-ban tanult konstruktorral. Ezzel állítok be egy kiindulási helyzetet az osztálynak, mely a példány létrehozásakor automatikusan meghívódik minden alkalommal. A bemeneti neuronok, rejtett neuronok és a kimeneti neuronok számát várja meghívásakor. Továbbá létrehoz egy „layers” és egy „activations” listát, melyekben megadom a mesterséges neurális hálóm rétegeinek összetételét. Az én esetemben ez két darab „*Dense*” réteg, egy „*ReLU*” aktivációs függvény és egy „*Softmax*” aktivációs függvény. Ezek működését a már korábban létrehozott fájlokban határoztam meg. Ezen funkcióknak is van saját előre definiált konstruktoruk, melyek előre beállítják például a „*Dense*” réteg esetén a súlyokat és az eltolásokat. A súlyok kezdő értékei, a már említett standard normál eloszlással generált véletlenszerű számok és az eltolások alap értéke nullára van állítva.

```python
import numpy as np

#ReLU activation function
class ReLU:
    def forward(self, _input):
        self.output = np.maximum(0, _input)

    def derivative(self, output):
        self.doutput = np.where(output > 0, 1, 0)

class Softmax:
    def forward(self, inputs):
        # exp_values = np.exp(inputs)
        #In case of exp_values has inf. in it
        exp_values = np.exp(inputs - inputs.max(axis=1, keepdims=True))
        probs = exp_values / np.sum(exp_values, axis=1, keepdims=True)

        self.output = probs

    def derivative(self, softmax):
        self.doutput = softmax * (1 - softmax)
```

*20. ábra: Aktivációs függvények*

```python
import numpy as np

class Dense:
    def __init__(self, n_input, n_neuron):
        self.weights = np.random.randn(n_input, n_neuron)
        self.biases = np.zeros((1, n_neuron))

        self.grad_weights = []
        self.grad_biases = []

    def forward(self, _input):
        self.output = np.dot(_input, self.weights) + self.biases
```

*21. ábra: Rejtett réteg*

#### 4.3.3.1 Forward propagation

Következő lépésként definiálom a „forward propagation” funkciót. Itt hívódik meg minden réteg és aktivációs függvény, melyet használok a jósolt értékek kiszámításához. Bemenetnek egy, már előre feldolgozott mondatokból álló listát vár.

Először az első „*Dense*” réteg „forward” metódusát használom. A bemenő értékek megszorzódnak a súlyokkal, majd hozzáadódnak az eltolások, ezzel megkapva a bemeneti értékét a következő rétegnek. A kapott eredményt az első aktivációs függvény, jelen esetben a *ReLU* módosítja. A *ReLU* minden negatív értéket nullával helyettesít, a pozitívakat pedig változatlanul hagyja. Ezen értékeket egy második „*Dense*” réteggel is módosítok. Ebben a rétegben már mások a generált súlyok. A számítások eredményét a „*Softmax*” nevű aktivációs függvénnyel átszámolva a jósolt értékeket kapom.

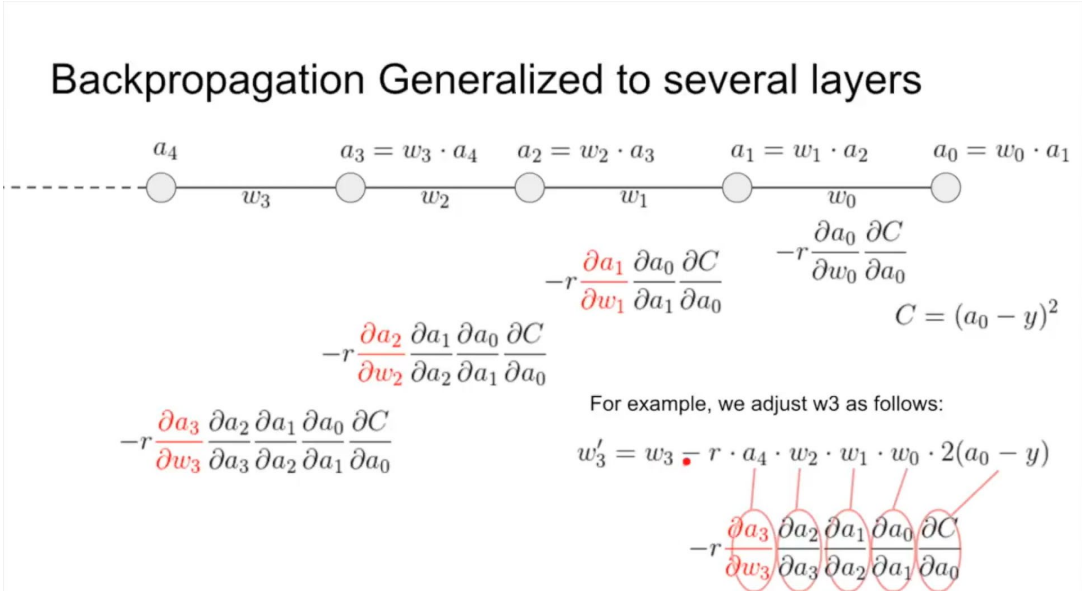
Itt megemlíteném a „Softmax” fontosságát, mivel van egy másik függvény is, az „Argmax”, amely nem megfelelő a neurális háló betanítására. A probléma, ami felmerül az utóbbi függvény esetében az az, hogy eredményül nem ad megfelelő mennyiségű információt. A kimenete kimerül annyiban, hogy 1-et rendel a legnagyobb értékű elem pozíciójához, a többihez 0-t. Ebből megtudjuk melyik válasz a legvalószínűbb, de nem derül ki, milyen mértékben biztos a programunk a válaszában. Pontosan erre jó a „Softmax”. Megtudjuk általa, melyik választ mekkora valószínűséggel gondolja megfelelő válasznak, ezzel fontos információkat biztosítva majd a „backward propagation” -nek. Ezért fontos a megfelelő aktivációs függvényt választani.

A „forward propagation” funkciónak a kimenete lesz ezen valószínűségek mátrixa.

#### *4.3.3.2 Backward propagation*

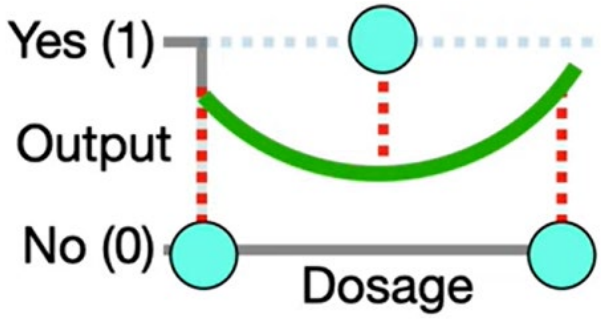
Az ebből kapott kimenet az első alkalommal egy teljesen véletlenszerű eredmény, általában a valószínűsége az elemeknek közel azonos szokott lenni ezen a ponton. Például, ha öt témaköröm van lehetséges eredménynek, akkor a várható első jóslatát a programnak így lehet elképzelni: [0.2, 0.2, 0.2, 0.2, 0.2]. Persze nem törvényszerű, hogy ilyen eredményt produkáljon, de a lényeg a tény, miszerint a program csak véletlenszerűen tippel.

Tegyük fel, hogy a program jóslata a következőképpen néz ki: [0.1, 0.4, 0.2, 0.2, 0.1], de a várt válaszunk így: [0, 0, 1, 0, 0]. Ez azt jelenti, a programunk rosszul jósolta meg a kimenetet. Szerinte a megfelelő válasz a második elem a listában, holott a várt eredmény a harmadik elem lenne. Itt válik hasznossá a „Softmax” funkció, mivel ki tudjuk pontosan számolni a távolságot a várt eredmény és a jósolt eredmény között.




*22. ábra: "Backpropagation" számítása a láncszabály segítségével [25]*

A „backward propagation” folyamatának fontos része a láncszabály, amely a többszörösen összetett függvények deriválását segíti elő. A fenti képen látható egy leegyszerűsített példa erre. A „C” jósolt és a várt érték különbségének a négyzetével egyezik meg, ezt hívjuk hiba értéknek. A példán a jósolt értéket „$a_0$” -val, a várt értéket pedig „y” -nal jelölik. A kimeneti réteg eredménye, amely a program által generált jóslat értéke. Ahhoz, hogy a „$w_3$” súlyt módosítsuk, a láncszabályt kell alkalmaznunk a rétegeken keresztül. Ez azt jelenti, hogy a hibaérték deriváltja az „$a_0$” deriváltjának vonatkozásában függ az „$a_0$” derivált osztva az „$a_1$” derivált értékével és ez lánc szerűen folytatódik visszafelé a „$w_3$” -ig. Ezt a képletet a lépésmérettel megszorozva kivonjuk a „$w_3$” -ból és megkapjuk az új „$w_3$” értéket. A „backward” funkció bemenetként előre feldolgozott mondatok listáját, a hozzá tartozó várt értékeket és a tanulási arányt várja.

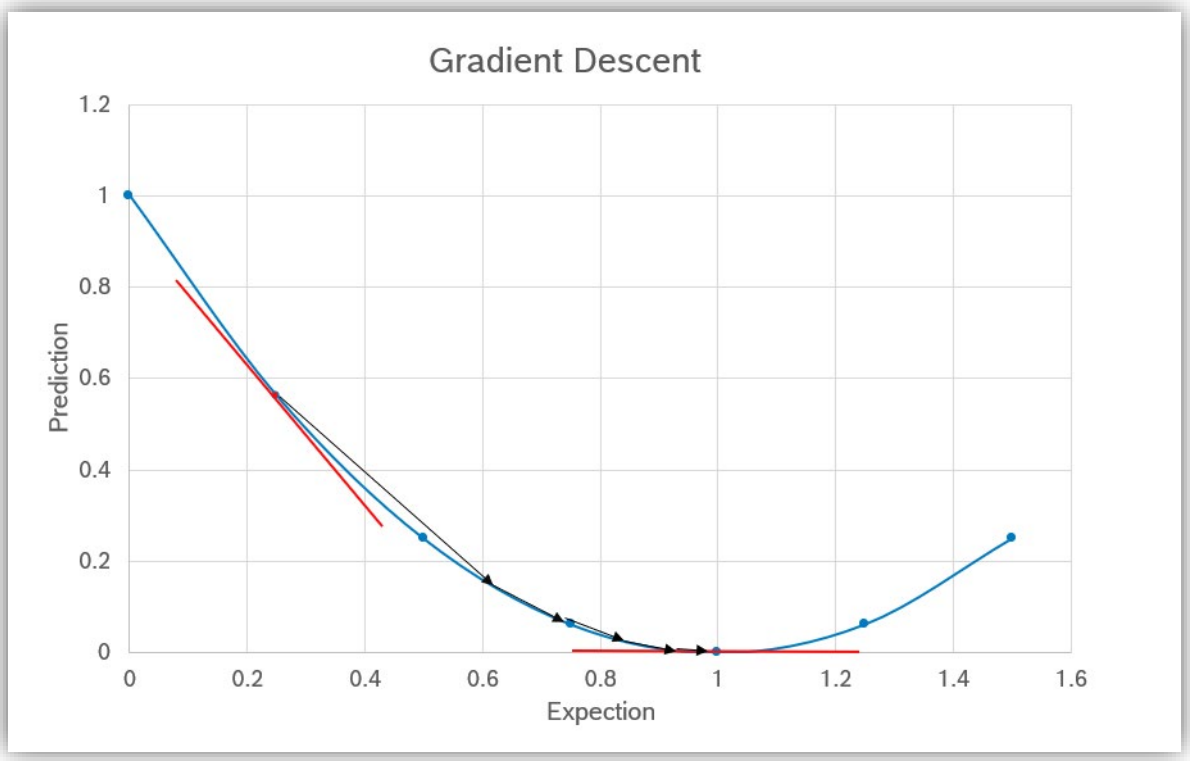



*23. ábra: Vizualizációs példa a hibára [26]*

A fenti képen látható pontok a példában a várt eredményt mutatják és a zöld görbe a jósolt függvényünk. A visszaterjesztés (backward propagation) lényege az, hogy az elvárt és a kapott eredmények közötti távolságot a lehető legkisebbre csökkentsük. Az ábrán ezek a távolságok piros szaggatott vonalakkal vannak jelölve. A hiba a jósolt eredményben úgy számítható ki, hogy a jóslatból kivonjuk a várt eredményt, így megkapva a két pont közötti távolságot. A visszaterjesztés számításait a „gradient descent” nevű veszteségfüggvény segítségével végzem el. Ez a függvény számos típusú probléma megoldásában hasznos tud lenni. Például a lineáris regresszióban, ahol a pontokra egy egyenest kell igazítani, keresve a lehető legkisebb összesített távolságot a pontoktól, így minden később megadott X értékre egy kétdimenziós ábrán meg tudjuk az egyenes segítségével jósolni a hozzá tartozó Y értéket. Emellett a logisztikus regresszióban is hasonlóképpen működik a megvalósítása, annyi eltéréssel, hogy itt nem egyenest, hanem egy görbét igazítunk az adatainkra. Végül kiemelném a klasztereket, ami lényegében bizonyos logikát követve meghatározott csoportokba osztja az adatainkat. Ezt úgy kell elképzelni, hogy ha egy kétdimenziós diagram pontjait nézzük és egy nagyobb mennyiségű adat egy kisebb területre csoportosul a többi adat pedig egy jól elkülöníthető másik pontnál van jelen, akkor ezeket az adatokat két klaszterbe csoportosíthatjuk. Ez legtöbbször adatelemzéseknél használatos.

A „gradient descent” a függvény meredekségének vizsgálatával változtatja az értékeinket. Szükség van hozzá a már korábban kiszámolt hiba négyzetértékére, melynek venni kell a deriváltját. A hiba visszaterjesztésével a hálózatban végig haladva minden súlyhoz és eltoláshoz kiszámoljuk mekkora mértékben járult hozzá a hibához. Ezzel megkapva a korrekciós értékeket.

A kapott gradiensek alapján frissítjük a súlyokat és az eltolásokat. Az alábbi ábrán látható, ahogy a „gradient descent” használatával a kezdő meredekség hogyan csökken. A tanulási aránnyal megszorozva pedig egyre kisebb léptékben változik az értékünk miközben közeledünk a várt értékhez.




*24. ábra: Gradient descent folyamatának vizualizációja*

A „backward” funkció ezeket a módszereket felhasználva számítja ki az új értékeket a mesterséges neurális háló súlyaihoz és eltolásaihoz.

Ezzel valójában a függvényünket akarjuk minél jobban és effektívebben a várt függvényre igazítani. Nagyobb probléma ennél a résznél fordult elő, mivel a rétegek közötti mátrix szorzások a dimenziók megfelelő kezelése nélkül nem működött. Ennek oka a funkció bonyolultságából adódott, melynek megoldására annak tüzetesebb körül járása, többfajta logikai megközelítése és két hét tesztelési folyamat elvégzése volt szükséges.

### 4.3.4 Training

A betanítás funkció bemenetként egy listát vár előre feldolgozott mondatokból, ezen mondatok várt válaszait, a tanulási arányt, a korszakok (epochs) számát (melyek jelentőségét lentebb fejtem ki) és a halom méretét (batch size) adom meg.

A korszakok száma valójában egy iteráció ciklusainak a számát határozza meg. Azt mutatja, hogy hányszor fusson végig az adataimon a betanítás. Erre azért van szükség, mert minél többször viszem át az adataimat a „forward” és a „backward” folyamatokon, annál több lehetősége lesz a programnak megközelítenie a várt

görbénket. A halom mérete azt adja meg, hogy az adataimból hány darabot szeretnék használni a betanításnál. Általában az adataim 80% -át használom betanításra, melyet véletlenszerű sorrendbe rakva, minden egyes betanításnál újra rendez. A maradék 20% -át később, a tesztelésnél használom fel. Ezt a 80-20% -os arány elfogadott és jól kitapasztalt a mesterséges neurális hálók betanításához. Ezen kívül használnak még 70-30-as arányt és olykor a 90-10-es is előfordul. Ez az arányszán nincs kőbe vésve, mindenki kitapasztalhatja a saját programjára vetítve, mi a legjobb hozzá. Az utolsó iteráció végén a kiszámított súlyokat és eltolásokat listává konvertálom.

A Pythonban két különböző adatformátum van, melyek szóba jöttek a modellem paramétereinek tárolása esetén. Ezek a „*pickle*” és a „*json*” formátum. Az én választásom az utóbbira esett több okból is. Elsősorban a *json* formátumot már használtuk több tantárgy keretén belül is, továbbá míg a *json* könnyen olvasható és értelmezhető, addig a *pickle* bináris formában menti el az adatokat. Az utóbbi tulajdonság előnyös a nagyobb adatmennyiségek továbbításánál, de hátránya, hogy a *json*-nel ellentétben nem biztonságos a kétes eredetű forrásból származó adatok kezeléséhez. Végezetül a *json* platformfüggetlen is, így nem volt kérdés, melyik adatformátumot használjam.

### 4.3.5 Teszt

Miután a modellem paramétereit elmentettem egy „model.json” nevű fájlba, már csak a teszt funkció maradt, ahol bemenetként az előre feldolgozott mondatokat tartalmazó listát, a hozzájuk tartozó válaszok listáját és a halom méretét várja.

Ebben a funkcióban kiválasztom az adataim maradék 20% -át és ezekkel még, a program számára ismeretlen adatokon végzem el a „forward propagation” -t.

Végül összehasonlítom a kapott eredményeket a vártakkal és egy osztást követően megkapom milyen pontossággal tud megjósolni egy a program által várhatóan értelmezhető mondatot.

További validációnak, megkértem néhány munkatársaimat, hogy utánam ők is teszteljék le a program pontosságát és használhatóságát. A visszajelzések a demó verzióra pozitívak voltak.

### 4.3.6 Chatbot megvalósítása

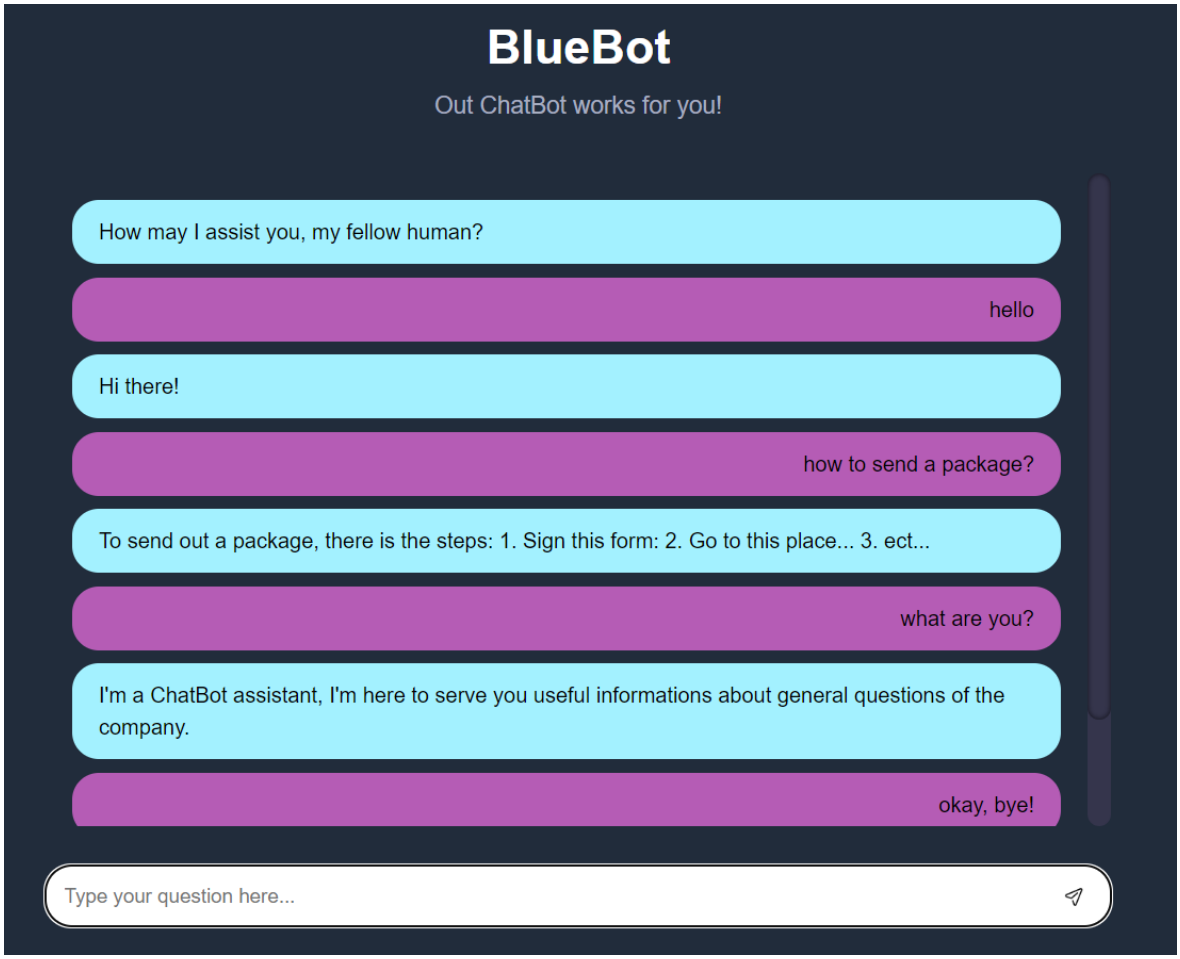
A chatbot minden eddig elmentett *json* fájlt beolvas, továbbá betölti a *spaCy* angol nyelvi modelljét is. A tesztelések alatt a „chatBot” funkció látja el a bemenetek fogadásának és a válaszok adásának szerepét, melyet később maga a webalkalmazás fogja átvenni. Itt egy végtelen ciklus vár egy bemenetet, melyre meghív egy funkcióból álló láncolatot, amely előfeldolgozza a bementet. Az előfeldolgozás itt is ugyan úgy folyik, mint a betanítás esetében. A mondatokat szavakra, a szavakat szótári alakjaikra bontja. Ezeket a „token” -eket összehasonlítja a program szótárában fellelhető szavakkal és ha talál egyezést, akkor a szótár adott szavának a helyére egy 1-est ír, ha nem, akkor 0-t. Végig iterál a szótár szavain az „enumerate()” funkcióval és minden bemeneti tokent megvizsgál. Eredményül egy a szótár hosszával megegyező bináris számokat tartalmazó vektort kapunk. Ezt az adatot már tudja értelmezni a programunk, tehát következő lépésként a „forward propagation” módszerével, a már megfelelően súlyozott és betanított mesterséges neurális háló modelljével kiszámoljuk a jósolt értéket, ami egy bináris értékeket tartalmazó vektor lesz. Ez a vektor mutatja mekkora valószínűséggel melyik témakör lehet az, amelyikről kérdeztünk.

Végül a „*pred_label*” funkció végig iterál a jósolt valószínűségeken és ha az nem ér el egy bizonyos küszöbértéket, akkor a végleges listába nem teszi bele. A végleges listát a valószínűségek értéke alapján csökkenő sorrendbe rendezi, így megkapva a legesélyesebb válaszok listáját. A választ a „*get_response*” függvénnyel kapja meg a chatbot, ahol az imént kapott lista első eleme alapján a megfelelő címkét kiválasztva visszaad egy véletlenszerű választ az adott címkéhez tartozó válaszlehetőségek közül (ha több van, mint egy darab).

### 4.3.7 WEB-alkalmazás megvalósítása

Ahogy már korábban szó volt róla, a webalkalmazás kontroller szerepét a *Flask* tölti be. A *Flask* jeleníti meg a *HTML* weboldalt a böngészőben. Mikor a felhasználó begépel egy kérdést és rákattint a küldés gombra vagy lenyomja az „Enter” billentyűt, akkor aktiválódik a *HTML* kódba integrált javascript „message” funkció. Ez a felhasználó bemenetét elmenti egy változóba, létrehoz egy *HTML* elemet, amit dinamikusan megjelenít a weboldalon, majd a *Flask* megkapja a „GET” kérést. Az üzenetet a chatbot funkcióin keresztül értelmezi és választ generál rá. A választ a „POST” kéréssel visszaadja a weboldalnak, ahol egy újabb *HTML* elem

létrehozásával megjeleníti a felhasználónak. A chatbot jelenlegi demó változata a *HTML* stílusának módosításához használt *CSS* kóddal az alábbi ábrán láthatóan néz ki.




*25. ábra: Chatbot webalkalmazás működés közben*

# 5 Összefoglalás

A szakdolgozatomban bemutatásra került a Bosch és a vállalaton belül egy automatizált megoldás arra, hogyan oszthatunk meg információt interaktív módon. Erre egy szabály alapú chatbotot találtam és arra a következtetésre jutottam, hogy egy hasonló megoldással készítek egy MI alapú chatbotot, amely az adminisztrációs feladatok megoldását segítené és kidolgozott, könnyen érthető válaszokat adna rá.

A célom ebben a szakdolgozatban az volt, hogy létrehozzak egy olyan működő chatbotot, amely adminisztrációs feladatokban és információ szerzésben jelentős mértékben megkönnyíti az emberek munkáját.

A program megvalósítása során több hasznos észrevétel született a fejlesztéseket illetően és ezek a fejlesztések lesznek implementálva a jövőben. A program jelenleg kevés adattal dolgozik melyhez egy *json* fájlban vannak a szükséges információk. A későbbiekben nagyban bővíteni szeretném a tudásbázist. Emellett a webalkalmazást elérhetővé fogjuk tenni először az osztály részére, majd az egész szervezet részére is. Ehhez viszont a chatbot tudását át kell helyezni egy adatbázisba, melyre a *mongoDB*-t fogom használni, a kielégítő szöveges adatkezelés okán. Tervezem az adatvédelem tekintetében, hogy az általam tervezett chatbot anonimizált módon tárolja az adatokat, hogy ne lehessen következtetni a felhasználó kilétére. Célom továbbá dinamikussá tenni a nyelvkezelést, tehát ha angol üzenetet küldünk neki, akkor angolul válaszol viszont, ha bemenetként egy magyarul írt kérdést fogalmazunk meg, akkor a válasz is magyar nyelven fog érkezni, függetlenül a korábbi kérdésektől. Tervben van, hogy az adott felhasználó által generált kérdések és azok válaszainak tárolását megoldjuk, megadva a lehetőséget a későbbi fejlesztéseknek. Ezen kívül meg fogom oldani a chatbot korábbi beszélgetéseire való referálás megértését is.

Összességében ezt egy nagyon hasznos és érdekes projektnek tartottam, melynek segítségével mélyen elmerülhettem egy szárnyait bontogató tudományág szerteágazó világában.

# 6 Summary

In my thesis, I presented Bosch and an automated solution for sharing interactive information within the company. I found a rule-based chatbot for this purpose and concluded that a similar solution could be developed using AI-based chatbot technology. This chatbot would assist with administrative tasks and provide well-developed, easy-to-understand answers.

The aim of my thesis was to create a functional chatbot that would significantly facilitate people's work in administrative tasks and information retrieval.

During the implementation of the programme, several useful observations were made regarding the development. These improvements will be implemented in the future. Currently, the programme works with limited data stored in a JSON file. In the future, I plan to expand the knowledge base significantly. In addition, we will make the web application available to the department and later to the whole organisation. However, this will require the chatbot's knowledge to be stored in a database, and I will be using MongoDB because of its satisfactory text data management capabilities. In terms of privacy, I plan to store the data in an anonymised form so that it cannot be linked to the identity of the users.

I also want to make the language processing more dynamic. If an English message is received, the chatbot will respond in English, and if a question is formulated in Hungarian, the response will be in Hungarian, regardless of previous questions. I also plan to store the user-generated questions and their corresponding answers, which will provide opportunities for future development. I will also ensure that the chatbot understands references to previous conversations.

Overall, I found this project to be very useful and interesting, allowing me to immerse myself in the diverse world of an emerging scientific field.

# 7 Irodalomjegyzék

[1] [Online]. Available: https://www.bosch.hu/vallalatunk/a-bosch-magyarorszagon/miskolc-robert-bosch-power-tool-kft/.

[2] [Online]. Available: https://www.bosch.com/stories/ai-chatbot-frizz/.

[3] [Online]. Available: https://onlim.com/en/the-history-of-chatbots/.

[4] Mitrović, Sandra, D. Andreoletti and O. Ayoub, Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text, arXiv preprint arXiv:2301.13852 , 2023.

[5] Athota, Lekha and et al, Chatbot for healthcare system using artificial intelligence, 8th International conference on reliability, infocom technologies and optimization (trends and future directions)(ICRITO), 2020.

[6] [Online]. Available: https://www.engati.com/blog/types-of-chatbots-and-their-applications.

[7] E. D. Liddy, Natural language processing, 2001.

[8] [Online]. Available: https://www.turing.com/kb/natural-language-processing-function-in-ai.

[9] [Online]. Available: https://www.unite.ai/10-best-python-libraries-for-natural-language-processing/.

[10] [Online]. Available: https://medium.datadriveninvestor.com/neural-networks-368d63e3bc56.

[11] E. Naqa, Issam and M. J. Martin, What is machine learning?, Springer International Publishing, 2015.

[12] [Online]. Available: https://www.ibm.com/topics/deep-learning.

[13] [Online]. Available: https://hackr.io/blog/flask-vs-django.

[14] Ghimire and Devndra, Comparative study on Python web frameworks: Flask and Django, 2020.

[15] [Online]. Available: https://www.spyder-ide.org/.

[16] [Online]. Available: https://www.python.org/.

[17] J. a. J. S. Keith, DOM scripting: Web design with Javascript and the document object model, Apress, 2011.

[18] Y. Vasiliev, Natural language processing with Python and spaCy: A practical introduction, No Starch Press, 2020.

[19] C. R. e. a. Harris, Array programming with NumPy, Nature 585.7825, 2020.

[20] S. Tosi, Matplotlib for Python developers, Packt Publishing Ltd, 2009.

[21] [Online]. Available: https://www.w3schools.com/statistics/statistics_standard_normal_distribution.php.

[22] [Online]. Available: https://www.geeksforgeeks.org/benefit-of-using-mvc/.

[23] [Online]. Available: https://spacy.io/usage.

[24] [Online]. Available: https://www.knime.com/blog/a-friendly-introduction-to-deep-neural-networks.

[25] [Online]. Available: https://www.youtube.com/watch?v=8d6jf7s6_Qs&t=436s.

[26] [Online]. Available: https://www.youtube.com/watch?v=iyn2zdALii8&t=346s.

[27] [Online]. Available: Neural Networks. Inspiration can come from anywhere, be… | by Ratik Puri | DataDrivenInvestor.

# 8 Melléklet

A CD mellékleten és az alábbi linken elérhető a webalkalmazás forráskódja, továbbá a feltűntetett fájlok: https://github.com/FallenPlan/szakdolgozat_TVIK4Ic

A link és a CD melléklet tartalma:

- CsomorBencePatrik_TVIK4I.docx
- CsomorBencePatrik_TVIK4I.pdf
- CsomorBencePatrik_TVIK4I_ osszefoglalas.docx
- CsomorBencePatrik_TVIK4I_osszefoglalas.pdf
- CsomorBencePatrik_TVIK4I_summary.docx
- CsomorBencePatrik_TVIK4I_summary.pdf
- Webalkalmazás forráskódja