



Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

36. OTDK Biológia Szekció

Genomikai nyelvmodellek bioinformatikai alkalmazásai a mikrobiológiában

Bodnár Babett

Molekuláris Bionika mérnöki BSc

4. évfolyam

2023

Témavezető: Dr. Ligeti Balázs

Tartalomjegyzék

1. Bevezetés	1
1.1. Szakmai háttér	1
1.2. A szakdolgozat felépítése	2
2. Előzmények	4
2.1. Genom, genomika	4
2.1.1. A genom	4
2.1.2. Genom annotáció	5
2.2. Mesterséges intelligencia	5
2.3. BERT modell	9
2.4. DNABERT	10
2.5. Kutatás jelentőségének indoklása, célkitűzés	11
3. Tervezés	13
3.1. Jupyter Notebook	13
3.2. Adathalmaz	13
3.2.1. Begyűjtés	13
3.2.2. Előfeldolgozás	14
3.3. Módosítások	15
3.3.1. BERT-DNABERT	15
3.3.2. DNABERT-ProkBERT	16
4. Tanítás	18
4.1. Architektúra	18
4.2. Tanítási folyamat	19
5. Eredmények	20
5.1. Előtanítás	20

5.2. Finomhangolás	23
6. Eredmények kiértékelése	26
6.1. Előtanítás	26
6.2. Finomhangolás	28
6.2.1. Kiértékelési mutatók	28
6.2.2. Tanítások eredményei	30
6.3. Validáció benchmarking adathalmazon	31
6.4. Konklúzió	32
7. Összefoglalás	33
7.1. Feladatok	33
7.2. Eredmények	33
7.3. Jövőbeli tervek	34
Felhasznált irodalom	36

1. Bevezetés

1.1. Szakmai háttér

Az elmúlt évtizedekben a bioinformatikai eszközök fejlődésével rengeteg új genomikai adat vált elérhetővé a kutatók számára. Az első sikeres genomszekvenálási módszerek az 1970-es évekből származnak, melyeket *first-generation sequencing* technikáknak nevezünk (Sanger-, Maxam és Gilbert-féle szekvenálás).[1] Ezeket még manuálisan végezték, így idő- és energia-igényesek voltak. 1987-ben azonban sikerült a folyamatot számítógép bevonásával automatizálni, így gyorsabbá tenni.[2] 1996-ban az új, párhuzamosított DNS-szekvenálási módszerek megjelenésével elérkezett az újgenerációs szekvenálás korszaka (NGS, Next-Generation Sequencing). 2005-től kezdve pedig a piacon is megjelentek az NGS-t lehetővé tevő platformok. A szekvenálási folyamat fejlődésével megkezdődött a teljes genomszekvenálás (WGS, Whole Genome Sequencing), azaz organizmusok csaknem teljes DNS-szekvenciájának meghatározása. Kezdetben vírusok és baktériumok, majd eukarióta szervezetek genomját tárták fel egyre gyorsuló ütemben.[3] Ezáltal hatalmas mennyiségű genomikai adat keletkezett, melyek feldolgozása és annotálása komplex, számításigényes feladat.

A genomika fejlődésével a mikrobiális közösségek vizsgálata is egyre nagyobb szerepet kap a kutatók között. Hatalmas metabolikus potenciáljuk képessé teszi őket komplex feladatok megoldására, úgymint szerves anyagok lebontása, kolonizáció, patogénekkal szembeni védelem. A mikrobiális közösségek így szerepet játszhatnak azoknak az új kihívásoknak a leküzdésében, amelyekkel a mai világban szembesülünk. Mind egészségügyi, mind ipari vonatkozásban sokrétűen segítségünkre lehetnek. Ahhoz, hogy működésüket megértsük és hasznunkra fordíthassuk, elsősorban a genomot kell megvizsgálnunk.

A rendelkezésünkre álló rengeteg genomikai adat megértésének igénye szükségessé tette az analízisükre irányuló algoritmusok kidolgozását és fejlesztését. A szekvenciák feldolgozásához elsősorban a gének annotációjára, címkézésére van szükség. A DNS-adatok ilyen célú vizsgálatára ma már többek között adatbázis-elméleten alapuló metódusokat, szövegkereső algoritmusokat és mesterséges intelligenciát alkalmaznak.[4]

A mikrobiom potenciáljának kiaknázásához a legújabban kutatott vizsgálati eszköz a mesterséges intelligencia. A hagyományos gépi tanulási keretrendszerek ebből a célból csak korlá-

tozottan alkalmazhatóak. A legtöbb tudományos vizsgálat ugyanis kis mintaszámmal történik ($N < 100$), a címkézett adat előállítása pedig költséges és időben elhúzódhat. Az adatok továbbá zajosak lehetnek, illetve a jellemzők száma jelentősen meghaladja a *feature*-ök számát. Előbbiek az adatok megszerzését és előkészítését, utóbbi jellemzők a gépi tanulás eredményességét nehezítik. Ezen limitációk áthidalására és a rendelkezésre álló adatok maximális kihasználására *transfer learning* alkalmazható, amely módszert transzformer modellel támogathatunk hatékonyan.

Munkám során korábbi kutatások megközelítéseit továbbgondolva, egy újszerű genomreprezentációt és kapcsolódó gépi tanulási keretrendszert dolgozok ki, amely az említett problémák kiküszöbölésére alkalmas. Céлом, hogy optimális módon tudjuk a meglévő tudásbázisunkat hasznosítani, amellyel így *transfer* és *few-shot learning* feladatokat oldhatunk meg.

1.2. A szakdolgozat felépítése

Dolgozatomban DNS szekvenciákat feldolgozó gépi tanulós módszereket vizsgállok. Feladatom során olyan modell fejlesztésén dolgozom, amely nem-humán genomi adatokon való előtanítás után, kevés címkézett adattal finomhangolva képes a kijelölt klasszifikációs feladatok elvégzésére. Céлом, hogy az egyetem adatain tanított modell versenyképes legyen hasonló feladatokra alkalmazott modellek között. Munkám menetét a következő fejezetekre bontva mutatom be.

2. Fejezet: Előzmények

Első lépésként a dolgozatomban megértéséhez szükséges alapokról írok, mind a biológiai, mind a számítástechnikai oldalról. Bemutatom a genomikai adatok felépítését és tulajdonságait, majd a DNS feldolgozásával kapcsolatos kihívásokat és ezek megoldására irányuló ötleteket ismertetem.

Ezután rátérek az számítástechnikai részre. A mesterséges intelligencia egy kifejezett ágával, a szekvenciális modellekkel foglalkozom: RNN, transzformer modellek. Ezen belül az utóbbi modelleket ismertetem jobban, megindokolva a jelen kérdéskörben való alkalmazhatóságukat. Ezután a munkámhoz kiválasztott és alkalmazott BERT modellt mutatom be, nagy hangsúlyt fektetve a genomikai adatokhoz módosított DNABERT változatára.

A DNABERT bemutatásával rá is térek a munkám során végzett irodalomkutatásra. Célom egy DNS-t feldolgozó modell kidolgozása, így a genomika és a bioinformatika, valamint a

számítástechnika genommal foglalkozó ágával kapcsolatban kutattam. Átnéztem az évek során alkotott és alkalmazott algoritmusokat, modelleket, valamint a megoldandó problémák kérdéskörét.

3. Fejezet: Tervezés

A harmadik fejezetben bemutatom a munkámhoz szükséges előkészületeimet. Írok a választott programozói felületről és a további szükséges alkalmazásokról. Ebben a fejezetben lesz szó a modellválasztásról, a rendelkezésre álló forrásokról, illetve az adathalmaz begyűjtéséről. Az adatok feldolgozásának módját és a kódbeli változtatásokat is itt részletezem.

4. Fejezet: Tanítás

A következő részben a modell tanításának tényleges menetéről írok. Kitérek az előtanítás, majd a finomhangolás feladatára is. Bemutatom a konfigurációs beállításokat és a paraméterezést a különböző tanításokhoz, valamint a rendelkezésre álló erőforrást és a futási időket.

5. Fejezet: Eredmények

Az ötödik fejezetben láthatóak külön-külön az előtanítás és a finomhangolás eredményei. Az eredményekhez tartozó magyarázatok és az adott ábrákra mutató linkek a 6. fejezetben találhatóak.

6. Fejezet: Kiértékelés

A hatodik fejezetben elemzem az elvégzett munkát és a kapott eredményeket. A kiértékelést egy összehasonlítással folytatom, amely során elhelyezem az elkészült modellt az elmúlt évek hasonló célú eszközei között egy benchmarking tesztalmazon való teljesítménye alapján. Végül leírom a konklúzióm az elvégzett munkáról és az eredményekről.

7. Fejezet: Összefoglalás

Az utolsó fejezetben összefoglalom a kutatásom. Leírom, mit sikerült elérni, és ez mennyiben felel meg az elvárásoknak. Javaslatot teszek változtatásra, amellyel egy rövid előretekintést is adok a projekt jövőjét illetően. Ezenkívül megjelölöm a kutatás távoli célját is, az alkalmazhatóságát a sikeres végkimenetel esetében.

2. Előzmények

Ebben a fejezetben ismertetem a munkám megértéséhez kapcsolódó szükséges előzetes információkat. Foglalkozom a genom felépítésével, a szekvenciák elemzésének feladatával és bemutatom a kapcsolatot a genomikai adat és a természetes emberi nyelv között. Ezután a mesterséges intelligencia DNS-szekvenciát elemző alkalmazásairól írok. Végül a témában olvasott és feldolgozott szakirodalmat mutatom be, nagy hangsúlyt fektetve arra a kutatásra, amelynek az eredményeképp létrejött a modell, amit alapul vettem a munkámhoz. Ez a DNABERT, egy DNS feldolgozására alakított BERT modell, amelyet humán genomon való előtanítás után, különböző feladatokra finomhangolták.

2.1. Genom, genomika

Jelen kutatáshoz fontos a genomika alapszintű ismerete. A következőkben a teljesség igénye nélkül írok a genom felépítéséről és vizsgálatáról.

2.1.1. A genom

Az élő szervezetek, pro- és eukarióták egyaránt, DNS-ben tárolják a genetikai információt. A DNS négyféle nukleotidból álló hosszú szekvencia. A nukleotidok kémiai egységek, melyeket egy-egy betűvel jelölünk: A, T, G és C. Ennek megfelelően egyetlen DNS-szálat leírhatunk ezen karakterek sorozataként.

A genom kifejezés egy organizmus teljes DNS-szekvenciáját jelöli. Genomszekvenálás során ez a szekvencia válik ismertté. A genom fehérjéket kódoló és egyéb, nem-kódoló DNS szakaszokból áll, amelyek pontos lokációja és funkciója azonban a szekvenálással nem derül ki. A szekvenciák teljes megértéséhez az első lépés a genomannotáció, melynek feladata ezen ismeretlen szakaszok megnevezése és funkciójának felderítése.[5] A genom annotációja azonban nehéz, nem rutinfeladat. A szekvenciák értelmezéséhez nem csupán adott szakaszok információi kellenek, hanem a kontextus is, amely a számítógépes algoritmusokkal történő elemzést tovább nehezíti. Ezt a tényt támasztják alá a humángenom-projekt utáni kutatási eredmények. A szekvencia feltárása nem jelentette a gének rövid időn belüli azonosítását.[6]

2.1.2. Genom annotáció

A genom annotációjára és génpredikcióra már számos módszer létezik. Korábban statisztikai elemzéssel, genetikai térképet létrehozva lehetett gének egymáshoz viszonyított helyzetét megállapítani. Mára a kutatások lehetővé tették gének funkciójának predikcióját csupán a szekvencia alapján.[7]

A módszereket két fő csoportra lehet bontani. Az első a hasonlóság alapú (*similarity based*) keresés. Ezek azon a feltételezésen alapszanak, hogy a funkcionális gének evolúciósan konzerváltabbak a nem-funkcionális géneknél, így szekvenciáik erősen hasonlítanak. A vizsgált és egy ismert gén között hasonlóságot felfedezve következtetni lehet tehát a vizsgált gén szerepére. Lokális és globális összehasonlító algoritmusok léteznek. Előbbire példaeszköz a BLAST programcsalád. Utóbbira újabban a CSTfinder szoftvert alkalmazzák, ezenkívül elterjedt a GeneWise és a PROCRUSTES.[8]

A második csoport az *ab initio* predikció, amely során a struktúrát használják gének detektálásához. A szekvenciákból két fajta információt olvasnak ki: rövid motívumokat, például *splicing*-helyek, start vagy stop kodonok, illetve a kodontartalmat, amely fajokra jellemző. Ezek a tartalmak a szekvenciák fő tulajdonságait adják meg. A módszerre példa algoritmusok a dinamikus programozás, lineáris diszkriminancia-analízis, *hidden markov model* (HMM) és a neurális hálók.[8]

Dolgozatomban egy neurális háló modellel való munkámat írom le, ezért a következőkben a mesterséges intelligencia genomikai vonatkozását mutatom be.

2.2. Mesterséges intelligencia

A genom megértésére irányuló, számítógépes algoritmusokon alapuló próbálkozások az 1980-as évek óta folyamatosan fejlődnek. A legfőbb alkalmazási területek ekkor még főleg a klasszifikáció (*classification*) és a csoportosítás (*clustering*), valamint a hasonlóság-mérés (*alignment*) és a mintázat-keresés (*pattern-mining*) voltak. Utóbbi kettő feladatra főleg statisztikán alapuló algoritmusokat dolgoztak ki. Neurális hálókat a 2000-es évek elejétől fogva alkalmaznak a klasszifikációs és csoportosítási feladatokra.[9] A modellek kezdetben a DNS egyes jellemzői alapján (*features*) számoltak. Nguyen és munkatársai 2016-ban megjelent cikkében egy új modellt mutatnak be, a konvolúciós neurális hálót (Convolutional Neuronal Network = CNN).[10] CNN-t használva a releváns jellemzők kiválasztása (*feature-selection, extraction*) már nem em-

beri feladat, hanem a modell teszi meg a bemeneti adatok alapján. Így az algoritmus hatékonyabban működik. Bosco és Di Gangi, szintén 2016-ban bizonyította mélytanuló algoritmusok segítségével, hogy a modellek valóban hatékonyan választják meg a szükséges jellemzőket.[11] Ezeknek a klasszifikációs feladatra alkalmazott konvolúciós hálóknak az egyik hátránya, hogy a bemeneti adatokat címkével (*label*) kell ellátni, amely költséges és hosszadalmas feladat. Ezenkívül a *feature extraction* továbbra is gondot jelent, a szekvenciáknak ugyanis nincs pontosan meghatározott, kiemelhető jellemzője.

Viszonylag új megközelítés az, hogy a genom analíziséhez szekvenciális inputot befogadó modellt alkalmazzanak. Az ilyen modellek a kapott adatokat nem független darabonként dolgozzák fel, adott jellemzők alapján. Ehelyett egyben, kontextussal együtt fogadják be, megőrizve a szekvenciák eredeti összefüggéseit. Példa ilyen adatfajtára az emberi szöveg, hangsorok, videók vagy egyéb, időtől függő adatsorok.

A DNS is összefüggő szekvencia, melynek egyes részeinek a megértéséhez szükség van a kontextusra is. A DNS azon tulajdonságából kiindulva, hogy a négy említett karakterrel leírható, hasonlítható a természetes emberi nyelvhez. Így alkalmazhatunk rá egy természetes nyelvet feldolgozó modellt.

A természetes nyelv modellezésén (NLP - *Natural Language Processing*) az 1950-es évektől kezdve dolgoznak kutatók. A különböző feladatokra kidolgozott algoritmusokat, mint például a szöveg folytatásának jóslása, fordítás, vagy szöveg összefoglalása, azóta folyamatosan kutatják és optimalizálják.[12] A legalkalmasabb modellek erre a célra ma a Recurrent Neural Network (RNN) és a transzformer modellek. Jellemzőjük, hogy nem csak az aktuális bemenettel számolnak, hanem megőrzik a korábban kapott adatokat is.

A transzformer modellek az adatfeldolgozást új módszerrel közelítik meg. Az ún. *attention* mechanizmus, amelyre épülnek, kiemeli a bemenet kulcsfontosságú részeit és az egyes részek közötti kapcsolatokat, így őrzi meg a bemenet kontextusát.[13][14]

A transzformer modell a neurális hálózatokhoz hasonlóan rétegekből épül fel. Két fő egysége az enkóder és dekóder, amelyekből egy modell mindig azonos mennyiségűt tartalmaz. Ezek további rétegekre bonthatók. Az egyes enkóderek felépítése teljesen megegyezik egymással: egy *self-attention* rétegből és egy *feed-forward* neurális hálóból (FNN) állnak. A *self-attention* réteg segíti a modellt abban, hogy az éppen vizsgált tokenhez (szó) a szekvencián belül akár távolabb elhelyezkedő, jelentésben azonban közel álló tokeneket rendeljen. Egyes modellek *multi-headed attention* mechanizmust használnak, így képesek több jellemző alapján mérni a

szavak közötti kapcsolatokat. Az *attention* rétegnek a kimenetét kapja egyetlen FNN, minden egyes pozícióra nézve. A dekóderek hasonlóan épülnek fel, azonban a két rész között egy *cross-attention* réteg segíti az enkóder kimenetét átadni a dekódernek.[15]

Az NLP modellekre jellemzően a transzformerek az első enkóder réteg előtt tokenizálják illetve vektorizálják az inputot. A tokenizálás során egy tokenizáló algoritmus a szavakat a modellhez rendelt szótár alapján indexekké alakítja. Ez a lépés után következik az ún. *embedding*. Az algoritmus a bemeneti szekvencia tokeneit adott hosszúságú vektorra alakítja, amellyel így jelentésüknek megfelelően tárolja az adott dimenziós térben. Az *embedding* során a hasonló jelentésű szavak hasonló reprezentációt, azaz a térben közeli elhelyezkedést kapnak. A szavak hasonlóságát egy előtanított *embedding* algoritmus tudja, a felhasználónak ezzel nincs feladata. Ezenkívül az *embedding* vektorokhoz tartozik egy-egy pozicionális vektor is, amely a szavak sorrendjét tárolja.[16]

A modell működésének megértéséhez az *attention* mechanizmust érdemes jobban megvizsgálni. Többféle *attention* mechanizmus létezik, melyek számítási módjukban eltérnek. A következőkben a többek között BERT modelleknél alkalmazott *scaled dot product attention* számítási módszert mutatom be.[17]

A számítás megértéséhez szükség van QKV (*Query – Key – Value*) mátrixmodell ismeretére is. Az *attention* rétegekben ezzel a három mátrixszal számol a modell, így kapja meg és tárolja a szavak kontextus mellett vett jelentését.

A Q, K, V mátrixokat az input szekvenciából kapjuk, a tanult W_q, W_k és W_v súlymátrixokkal való szorzással. Ezeket a mátrixokat a következőképpen lehet elképzelni. A *query* mint vektor az input egyetlen szavának *embedding* vektora, a "lekérdezés". A *Query* mátrix az input minden egyes szavának *embedding* vektora, konkatenálva - lekérdezések egymás után. A *Key* mátrix a modell céljától függő "kulcs", amely szintén szavak *embedding* vektorát tartalmazza, a *Query*-hez hasonlóan mátrix formában tárolva. Ez utóbbi egy adatbázisként képzelhető el. Ehhez hasonlítja a modell a *Query* egyes vektorait. A *query*-k és a *Key* minden elemének "hasonlósága" adja meg az *attention score*-t, azaz a hasonlóságra vonatkozó pontot. A harmadik, *Value* mátrix a *Key* mátrix "párja", amely a bemenet szavainak *embedding* vektor formáját tartalmazza. Ez minden egyes *attention* lépésben frissül az előző két mátrix hasonlósága alapján. A *Value* legvégül a kontextussal összeegyeztetett jelentésben tárolja az *embedding* vektorokat, amelyeket így már *attention* vektoroknak hívunk.

Matematikailag a következőképp néz ki a *scaled dot product attention* egyenlete:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{D_k}})V = AV \quad (2.1)$$

Ahol D_k a *Key* dimenzióját, A az *attention score*-t vagy mátrixot jelöli, Q , K és V pedig a már ismertetett mátrixok.

Az egyenlet és a három mátrix matematikai levezetése pedig a következő.

A *query* és a *Key* hasonlóságát a skaláris szorzatuk adja meg, ez a QK^T . A túl nagy értékek elkerülése érdekében le kell osztani az eredményt a vektorok dimenziójának négyzetgyökével. Ez a $\sqrt{D_k}$ -val való osztás így segíti az eltűnő grádiens (*vanishing gradient*) kiküszöbölését. A mechanizmus egyetlen nem-lineáris lépése a *softmax* aktivációs függvény alkalmazása a $\frac{QK^T}{\sqrt{D_k}}$ mátrix minden oszlopára. A funkció exponenciálisan növeli a nagy értékeket, míg a kicsi vagy negatív értékeket nullához közelre csökkenti. A függvény továbbá normalizálja az oszlopokat úgy, hogy azokon belül az értékek összege egyet adjon ki. Az eddigi lépésekkel kapjuk az A , azaz *attention* mátrixot. Ennek végül V -vel való lineáris kombinációját véve a bemenet szavainak új *embedding* vektorait, a kontextuális reprezentációjukat kapjuk.

Az *attention* mechanizmusokat a számítási módjukon kívül csoportosíthatjuk a Q és K , V mátrixok bemenetének származása alapján is. Így a *self* – *attention* mechanizmus bemenete mindig az előző réteg kimenete; a *cross-attention query* vektorait a dekóder, K , V mátrixokat pedig az enkóder kimeneteiből kapjuk. A *masked self-attention* során a *query* vektorokkal való számításokat csak az aktuális bemenetig végzi a modell, nem hasonlít össze minden lehetséges *query-key* párost. A *multi-headed attention* többféle W_q , W_k és W_v súlymátrixokat tanul, így a szavak közötti kapcsolatokat több szempontból vizsgálja. A *multi-headed attention* egyenlete a következő:

$$MultiHead(Q, K, V) = Concat(h_1, h_2, \dots, h_h)W^O \quad (2.2)$$

Ahol h_i az egyes *attention* fejeket jelöli, W^O pedig az egyenletre vonatkozó tanult súlymátrix.[13][14] A transzformer modellek további összetevőit ebben a dolgozatban nem fejtem ki, a kutatás megértéséhez kevésbé szükségesek. Az említett FNN rétegen kívül *residual connection* segítségével mélyítik a transzformer modelleket. Ezenkívül az egyes rétegek között további normalizációs, lineáris és softmax funkciókat ellátó rétegek helyezkednek el. Ezek feladata a számítások eredményeinek skálázása, illetve tartományokon belül tartása.

A transzformer architektúrát és az *attention* mechanizmust részletesebben az eredeti cikk, *Attention is all you need* írja le.[15] Munkám megértéséhez az olvasónak további ismeretek

nem szükségesek a modell felépítéséről, így dolgozatomban nem célok annak részletesebb ismertetése.

2.3. BERT modell

Az *attention* mechanizmus 2017-es megjelenése óta a transzformer modellek rengeteg új változata jött létre. Munkám során a Bidirectional Encoder Representations from Transformers, azaz BERT modellel dolgozom.[18]

A BERT architektúra csak enkódereket tartalmaz, dekódereket nem. Ezáltal képes a kétirányúságra: a bemenő adat előtti és utáni kontextust is figyelembe veszi. A korábbi transzformerek csak a "baloldali" adatokkal való kapcsolatot figyelték. A BERT felépítése a dekóderek hiányát leszámítva megegyezik a korábban leírt transzformer architektúrával.

A BERT modellek tanítása két részből áll, előtanítás és finomhangolás.

Az előtanítás során két alapfeladatra tanítják a modellt, amelyekre épülnek a finomhangolás feladatai. Ezek a BERT esetében a maszkolt nyelvmódel (Masked Language Model, MLM) és a következő-mondat predikció (Next Sentence Prediction, NSP) feladatok. A tanító halmaz rengeteg címkézetlen szöveges adat. MLM feladat során egyes szavak maszkolásával tanítják a modellt arra, hogy képes legyen a mondatban egy hiányzó szót megmondani. Az NSP feladatra párokba rendezett mondatokon tanul a modell, így a köztük lévő kapcsolatot, vagy éppen annak hiányát tanulja meg.

A finomhangolás során a már előtanított paramétereket finomítják további feladatokra, kevés címkézett adattal. Ez a technika egyfajta transfer-learning: a finomhangoláshoz már előtanított paraméterekkel, súlyokkal rendelkezik a modell. Maga a transfer-learning az a módszer, amellyel egy már betanított modell tudását használják fel egy új, hasonló vonatkozású modell tanításához. Az NLP-ben ez azt jelenti, hogy a modell tanítása után az utolsó, feladatot végző réteget egy új, másik feladatra való réteggel helyettesítik és ezután már csak ezt a réteget tanítják. A módszerrel hatékonyan csökkentik a tanításra fordított időt, ugyanis egy architektúra esetén minden egyes finomhangolási feladat ugyanazt az egy előtanított modellt veheti alapul. A kevés címkézett adat által fellépő gond is megoldhatónak tűnik ezzel a technikával, hiszen a finomhangoláshoz viszonylag kevés adat szükséges.

A BERT modellt egy 2021-ben megjelent cikkben új vonatkozásban mutatták be a kutatók: a humán genom feldolgozására alakították át, ennek megfelelően DNABERT-nek nevezték

2.4. DNABERT

A DNABERT a BERT modell egy módosított változata, amely genomi adatok feldolgozására alkalmas. A kutatás célja egy olyan modell létrehozása volt, amely egyszerre képes a szekvenciák általános megértésére, a kontextus figyelembevételére és klasszifikációra limitált címkézett adatok alapján. A szerzők továbbá a genom nem-kódoló, szabályozó régiószakaszainak osztályozását tűzték ki célul.[18]

A biológiai áttekintésben megnevezett annotációs problémák tovább bonyolódnak, amikor a szabályozó régiók megértéséről van szó. Ezek a szakaszok ugyanis a fehérjekódoló részekkel szemben nem egyértelműek és univerzálisak. Nem csak fajok és organizmusok, de akár sejtek között is eltérhet bizonyos gének aktivitása, ebből következtethetően a szabályozó régiójuk működése változó.[20] A genom ezen szabályozó szakaszát CRE-nek (cis-regulatory element) nevezzük. A CRE működése komplex: befolyásolhatja a kromatin elérhetősége, a transzkripció faktorok kötődése, metiláció, hiszton-modifikáció és az interakciók más szakaszokkal.[21] A DNABERT megalkotásával az input kontextusának megtartása lehetőséget ad arra, hogy a szakasz távoli részei közötti összefüggéseket is értse a modell. Korábbi kutatások (CNN, RNN modellek) nem érték el a DNABERT megalkotói által kitűzött célokat, vagy kevés címkézett adat állt rendelkezésükre. A DNABERT egy újszerű megközelítése a problémának, amelyre teljesülnek a kutatók kitételei.

A DNABERT-nek tanítóanyaga a humán genom. A DNS-szekvenciát k -merekre, azaz $k \in 3, 4, 5, 6$ hosszúságú, átfedő tokenekre bontották fel. Az átfedő k -mer alapú felbontással csökkenthető a DNS felbontása által esetlegesen fellépő befolyásolása a tárolt információnak. Az adatokat továbbá maximum 510 token (510 nukleotid) hosszúságú blokkokra bontották, a modell maximum befogadóképessége miatt.

A transzformer-architektúra megegyezik az eredeti BERT *base* beállításával. A DNABERT az AdamW *optimizer*-t az alábbi paraméterekkel alkalmazta: $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1e^{-6}$, $\text{weight decay} = 0.01$.

A modellben továbbá egy fontos változtatás a maszkolási feladat átírása, amelyet a k -merekhez kellett igazítani. Egy-egy token maszkolásával ugyanis az előtte és utána következő tokenekből, az átfedőség miatt, következtetni lehet a hiányzó adatra. A modell célja ennél mé-

lyebb összefüggések keresése a genomban, így a DNABERT maszkolási feladata k értékének megfelelően k egymás utáni token maszkolását jelenti.

A DNABERT modell a *pretraining* során az első 100000 lépésig az input 15%-át, a maradék 20000 lépésben a 20%-át maszkolva tanult. Ez a félig-felügyelt tanulási mód teszi lehetővé a DNS szekvenciák mélyebb megértését, így a modelleket viszonylag kevés címkézett adattal lehet további feladatokra tanítani. A finomhangolás a DNABERT modell esetében egy új, klasszifikációs réteg tanítását jelenti, amely az előtanítás súlyait optimalizálja. A DNABERT modellt az eredeti kutatás során a következő feladatokra tanították: promóter predikció, transzkripció faktor kötőhely (TFBS) predikció, *splice donor* és akceptor helyek predikciója. Ezenkívül a szekvenciában található, nagy jelentőségű DNS-szakaszok, motívumok vizualizációját is lehetővé tették *attention score*-ok és a szekvencia mentén ezen pontok eloszlása alapján. A DNABERT kutatói továbbá vizsgálták az SNP-k előfordulásának hatásait a predikciókra.

2.5. Kutatás jelentőségének indoklása, célkitűzés

Munkám során egy újszerű genomreprezentációt és kapcsolódó gépi tanulási keretrendszert dolgozok ki a DNABERT modellt felhasználva és átalakítva. Célom a modell mikrobiom adatokon való előtanítása majd finomhangolása nagy jelentőségű bioinformatikai problémákra, mint például a promóter-helyek felismerése. A bevezetésben ismertetett problémák miatt fontos, hogy meglévő tudásbázisunkat optimális módon használhassuk fel. A BERT modell DNS-re alakított változata potenciális megoldást nyújthat a mikrobiom adatok megértésére, a *transfer learning* és az ezáltal elérhető *few-shot* feladatok lehetőségével.

A DNABERT sikeres tanítása és publikálása óta a modellt több kutatáshoz is felhasználták. A *huggingface*-en elérhető eredmények mind eukarióta, többségében emlős genomot használtak az eredeti előtanított modell finomhangolására. Egyéb kutatás alatt ugyan prokarióta szekvenciák analízisével foglalkoznak, azonban ehhez továbbra is a humán genomon előtanított modellt alkalmazzák.[22]

A mikrobiológia prokarióta genommal foglalkozik, a mikrobiommal. A mikrobiom kutatásának jelentőségéről a bevezetésben volt szó. Ezen adatokat meg kell különböztetni az eukarióta adatoktól, a prokarióta és eukarióta genom két külön struktúrának felel meg.

A két genom struktúrája közötti különbséggel is foglalkoznak a kutatók. Ugyan mindkettő szerepe a genetikai információ tárolása, transzkripció majd transzláció, ezt eltérő módon teszik

meg. A következőkben kiemelek néhány megemlíthető különbséget a teljesség igénye nélkül.

A prokarióta szervezetek egyik fontos tulajdonsága, hogy génjeikhez policisztronos operon tartozhat. Ehhez a szabályozó régióhoz több gén tartozhat, amelyek funkciója kapcsolatban áll egymással. Ezekért a gén-klaszterekért így egy promóter felel. Ezenfelül a genomjuk kompaktabb, tömörebben tartalmaz információt: a gének leolvasási kerete akár át is fedhet a szekvenciában. Fontos megemlíteni a gének olvasási irányát is. A prokarióta organizmusokban ugyanis a DNS egyik-másik száláról való olvasás nem ugyanazt jelenti.

Az eukarióta szervezetekben egy-egy szabályozó régió egyetlen génhez tartozik, tehát minden génnek megvan a saját promótere és szabályozó eleme. Míg a prokarióták génjei akár át is fedhetnek, az eukarióta géneket nem-kódoló intron szakaszok szakíthatják meg. Ezek mentén a szekvenciát többféleképpen lehet darabolni (splicing), így az többféle eltérő fehérjét kódolhat.

A transzkripció mechanizmusát ismerni kiemelkedően fontos, megértésük ugyanis kulcsfontosságú az akár kutatási célú alkalmazásukhoz.[23] A transzkripció kezdetének helye (Transcription Start Site, TSS) körül is fontos különbségeket figyeltek meg a kutatók eukarióták és prokarióták között. A promóterek a gének előtt található rövid szabályozó szakaszok, amelyeknek a transzkripcióban van szerepük. A prokarióta promóterek két fő motívumot tartalmaznak, -10 és -35 pozíciókban. Az eukarióta promóterek ezzel szemben több motívumból állnak, komplexebbek. Ezek a rövid szakaszok azonban degeneratívak, kevés egyezik meg teljes mértékben a konszenzus szekvenciával. Megtalálásuk így nehéz és nem egyértelmű feladat.[24]

A promóterekhez kötődő, a két struktúra transzkripcióját szabályozó fehérjék is különbözőek. Kutatók összehasonlították az eukarióta és prokarióta transzkripcióban részt vevő aminosav-szekvenciáját a DNS-kötő fehérjéknek és a transzkripciós faktoroknak. Az eredmények azt mutatták, hogy az aminosav- és aminosav-csoport-gyakoriság eltérő mintázatot adnak a két szervezetben, ezek vizsgálatával egyértelműen megkülönböztethetőek.[25]

Mindezen különbségeket ismerve és további, ismeretleneket feltételezve megalapozott az a gondolat, hogy a prokarióta és eukarióta genom vizsgálatára külön modellt érdemes tanítani. A releváns kutatásokat áttekintve a DNABERT-hez hasonló felépítésű, a szekvenciát általánosan megértő algoritmust, amely kevés címkézett adattal is jól tanítható és prokariótákat vizsgál, nem találtam. Munkám ezért a prokarióta genom, főleg a mikrobiomot vizsgáló kutatások között újszerű megközelítést alkalmaz. A saját adatokon előtanított majd alfeladatokra finomhangolt modell célja, hogy validálás után, a hasonló kutatások között elfogadható, akár jobb eredményeket is hozzon.

3. Tervezés

Ebben a fejezetben a modell tanításához szükséges előzetes lépéseket írom le. A használt felületről, a kódok futtatásához szükséges keretrendszer megteremtéséről lesz szó elsőként. Ezután mind az előtanítás, mind a finomhangolás adathalmazának összeállításáról és előfeldolgozásáról írok.

3.1. Jupyter Notebook

Munkámhoz a Jupyter Notebook ingyenesen használható, böngésző alapú interaktív munkakörnyezetet használtam. Az egyetem Jupyter szerverét az egyetemi felhasználómon keresztül ssh szerver-kliens kapcsolattal értem el. Az előzményekben bemutatott DNABERT, azaz DNS feldolgozására átalakított BERT modell kódjával dolgoztam. Mivel a tanítás idő- és erőforrásigényes, munkám során a DNABERT által vizsgált k -merek közül csak eggyel, a $k = 6$ felosztással dolgoztam.

3.2. Adathalmaz

Az előtanításhoz rendelkezésre álló tanítóanyag az emberi bél-mikrobióta. A finomhangolásra a PPD (Prokaryotic Promoter Database) oldaláról letöltött, kísérletekkel ellenőrzött prokarióta promóter adatokat dolgoztam fel. A klasszifikációs feladathoz a negatív mintát egy negatív minta-generátorral készítettem.

3.2.1. Begyűjtés

Előtanítás A humán bél-mikrobiom adatokat 4744 rendezett fasta fájlból gyűjtöttem össze, amelyek összesen 697011 szekvenciát tartalmaznak. Az adatok az OTKA PD 138055 projekt során létrejött adatbázisból származnak. A genomok között a legrövidebb szekvencia 105, míg a leghosszabb 6802557 nukleotid hosszú volt. Összesen 10.7 GB kigyűjtött, feldolgozatlan tanítóanyagom volt.

Finomhangolás A finomhangoláshoz összesen 129149 promóterszekvenciát töltöttem le a PPD-ről. Ebből az E. coli-ból származó, illetve az ismétlődő szekvenciák lettek szűrve, valamint azok, amelyek nem voltak legalább 80 nukleotid hosszúak. Ezen felül azokat a szekvenciákat kellett eltávolítani, amelyek a későbbiekben szereplő validációs adathalmazban szerepeltek. Így végül összesen 115326 darab valódi promóter szekvencia állt rendelkezésre.

A finomhangoláshoz továbbá a promótereken kívül negatív minta, azaz nem-promóter szekvencia is szükséges. Ez nehéz feladat, ugyanis a modell a "túl könnyű" feladattal túlilleszkedhet, egyéb nem jól megválasztott mintákkal pedig rossz tanulási eredményt érhet el.

Ehhez Murilo Henrique Anzolini Cassiano és Rafael Silva-Rocha kutatók negatív minta-generátorával dolgoztam. Ők ezzel a programmal dolgoztak a különböző promóterpredikációs algoritmusok képességeinek összehasonlítása során.[26] A kód egy adott nevű fájlt olvas be, ez tartalmazza a promóter szekvenciákat. A promóterekben megszámolja az egyes nukleotidokat, majd ezek arányai alapján generál új szekvenciákat. A negatív minták hosszát és mennyiségét tetszőlegesen beállíthatjuk. A nukleotidok aránya fajspecifikus lehet, ezért a negatív minták generálását baktériumonként kellett elvégezni, a különböző fajoknak megfelelő nukleotid-arányokat beolvasva. A kódot ezenkívül kiegészítettem a pozitív mintáknál is alkalmazott, ismétlődést szűrő kódrésszel. A pozitív mintákkal megegyező számú, tehát 115326 darab negatív minta lett.

3.2.2. Előfeldolgozás

Az adatokkal a legelső feladatom az ismétlődő szekvenciák jelenlétének kiküszöbölése.

A DNS szekvenciákat a DNABERT modellhez hasonlóan átfedő k -mer formában célszerű feldolgozni, mind az előtanítás, mind a finomhangolás esetében. Az első feladatom így a k -merizálás. A k -mer forma a következőképpen néz ki. Egy adott CCTGTACTA szekvencia $k = 6$ esetében, átfedéssel CCTGTA CTGTAC TGTACT GTACTA lesz. A szekvencián tehát egy 6 karakter hosszú ablakot léptetünk nukleotidonként és minden lépésben kiírjuk az ablakban látható részletet.

Előtanítás Az előtanítás eredetileg 10.7 GB adataiból a k -merizálás után 73.5 GB adat lett. A szekvenciákat ezután még random módon összekevertem, hogy a tanítóanyag bármilyen jellegű rendezettségét elvegyem. Végül 80%-20% arányban felosztottam az adathalmazt tanító és teszt halmazokra.

Finomhangolás A finomhangolás szekvenciáit a k -merizáláson felül a klasszifikációs feladathoz címkézni is kellett. Mind a pozitív, promóter szekvenciákat, mind a negatív mintákat elláttam a megfelelő címkékkel (1 és 0), majd az előző adathalmazhoz hasonlóan randomizáltam a sorrendet. A címkézett adatokat '*sequence*', '*label*' header-rel láttam el és *.csv* fájlban tároltam. A finomhangolás adatait az előtanításhoz hasonlóan 80%-20% felosztásban mentettem el tanító és teszt halmazokra. Ahhoz, hogy a felosztás után is megmaradjon a pozitív-negatív minták egyenlő aránya, előbb bontottam fel ebben az arányban a pozitív és negatív minták fájljait, majd ezeket a részhalmazokat kevertem egymással.

Tokenizálás Mivel az előtanításhoz való tanítóanyag hatalmas, a tanítást pedig többféle paraméterezéssel is elindítottam, az előtanítás esetében különbontottam az input szekvenciák tokenizálásának lépését a tanítástól. Így ezt nem kellett minden alkalommal elvégeznie a modellnek, rengeteg idővel le tudtam rövidíteni a tanítás folyamatát.

A tokenizálás a szavak egy adott szótár alapján történő átalakítása indexekké. A szótár jelen esetben a $k = 6$ -mer szavakat tartalmazza, azaz a négy nukleotid karakter minden lehetséges permutációja 6 helyen. Összesen $4^6 = 4096$ ilyen k -mer létezik. Ezenfelül 5 egyéb token van a szótárban, amelyek a tanításhoz kellenek: [CLS], [SEP], [MASK], [PAD] és [UNK]. Minden tokenizált mondat elején egy [CLS] token található, amely utal a mondat tartalmára. A mondatok végét a [SEP] jelzi. A [MASK] token a maszkolási feladat tanításához szükséges. A [PAD] és az [UNK] tokenek rendre a padding-hez és az ismeretlen szavak jelzésére vannak. Így összesen 4101 tagból áll a 6-merek szótára.

3.3. Módosítások

Ebben a részben részletesebben kifejtem a BERT modellen végzett módosításokat.

3.3.1. BERT-DNABERT

A BERT és a DNABERT tanítása közötti fő különbség az előtanítás maszkolási feladata. A kód mélyebb megértéséhez átnéztem a maszkolás folyamatának DNS-re alakított változatát.

A BERT modell maszkolási feladata a következőképpen néz ki. A tanításhoz megadott paraméternek megfelelő mennyiségű tokent választ a modell random módon a maszkolásra. Ezek 80%-át [MASK] tokennel, 10%-át random tokennel és 10%-át az eredetivel megegyező

tokennel helyettesít.

A DNABERT szerzői ezt a feladatot alakították át a k -merekhez illesztve. Ahelyett, hogy 1-1 szót maszkoltak volna, k egymás utáni tokennel tették ezt. A modell először a paraméterezésnek megfelelően kiválasztotta a maszkolni kívánt tokeneket. Ezután a kód módosításának eleget téve az előtte és utána következő k -merek is hozzávette a "maszkolandó" tokenekhez. A kód írásakor az indexelési hibák elkerülése érdekében figyeltek arra, hogy a szekvenciák elején és a végén a megfelelő módon legyenek kiválasztva az indexek.

3.3.2. DNABERT-ProkBERT

A DNABERT modell kódján a munkámhoz egy kevés módosítást kellett elvégezniem.

A DNABERT előtanító kódja kétféle módon dolgozhatja fel az inputot. Mindkettő esetben a tanító adat előfeldolgozott változata egy byte-stream-ben lesz végül tárolva, hogy a későbbiekben könnyen elérje a modell a kész inputot. Ezen funkciók hátránya azonban, hogy míg pár GB-nyi adattal hatékonyan dolgoznak, több 10 Gb adatra *out - of - memory* hibaüzenetet ad. Ennek a problémának a kiküszöbölésére egy új módszert kellett bevezetnem az adatok betöltéséhez.

Az eredeti Dataset osztály az egész input adatot betölti egyetlen változóba, majd ezen fut végig miközben tokenizálja és blokkokra bontja. Ahogy a 3.3.2-es fejezet Tokenizálás című bekezdésében írtam, a tanításonkénti tokenizálás helyett ezt a lépést előre elvégeztem és egy új szöveges fájlba mentettem az előfeldolgozott inputot. Ez azonban csak az idő rövidítésén segít, az *out - of - memory* problémára nem megoldás, az adathalmaz továbbra is hatalmas. Egy olyan módszert kerestem tehát, amely nem tölti be változóba a tanító halmazt, vagy nem egyben teszi meg.

Az IterableDataset lett végül az az osztály, amellyel megoldottam a nagy adathalmazok problémáját. Az IterableDataset ugyanis nem tölti be az egész adathalmazt, hanem csak egy iterátorral tér vissza. Így akármennyi adattal dolgozik a modell, mivel nem tölti be változóba, nem ütközik memóriaproblémákba.

Az osztálynak definiálnom kellett egy "hossz" tulajdonságot, mivel az iterátor típusra ilyen nincs értelmezve. A hosszt az adathalmazt tároló fájl megnyitva egy enumerációs függvénnyel számíttattam ki, ezt találtam a leggyorsabb megoldásnak rá. Ezenkívül az iterátor által mutatott elemet, a tokenizált és blokkosított szekvenciát át kellett alakítani, mivel a szöveges dokumentumban *string*-ként volt tárolva. Így az előfeldolgozás egyetlen lépése ezzel az osztállyal egy

$int(x)$ konverzió elvégzése.

Az eredeti kódban az ellenőrzőpontok alapján való tanítás számításai néhány próba majd utánaszámolás alapján nem működtek rendeltetésszerűen, ezért a kód ezen részén is változtattam. A tanítás lépésekre van bontva, egy lépés pedig egy batch-nek megfelelő iterációszámot jelent. Tanítás közben az ellenőrzőpontok nevei *checkpoint* – *"num_of_steps"* minta alapján lettek elmentve. Betöltéskor azonban az iterációszám, ahonnan folytatni kell a tanítást, nem a *"num_of_steps"* és a batch-méret szorzataként lett számolva, hanem csak a *"num_of_steps"*-ként. Így minden, ellenőrzőponttól folytatott tanítás szinte a folyamat elejétől indult a valódi iterációszám helyett. Ezért kellett megtennem itt a kód átírását.

4. Tanítás

Ebben a fejezetben bemutatom a DNABERT saját adatokon való előtanításának és finomhangolásának futását. Az architektúra és paraméterezés fontosabb változtatásait említem meg.

4.1. Architektúra

A transzformer modell esetében a modell architektúrát és a tokenizáló beállításait is érdemes megvizsgálni. A DNABERT modell konfigurációs fájlja, *config.json*, a forrás $k = 6$ -mer mappájában található. Az architektúra neve *BertForMaskedLM*. A fájlban az attention mechanizmusra vonatkozó értékeket (fejek, rétegek száma), rétegszámokat, méreteket, aktivációs függvényt lehet megválasztani. Ezek az értékek befolyásolják a modell méretét, a tanulás paramétereinek számát. Ezek pedig az NLP területén jelentősek a modell teljesítményét tekintve.[27]

Az architektúra értékei közül az *attention* fejek számát változtattam meg. Az eredetileg 12 fejből álló modell helyett 8 fejet alkalmaztam. Céloom ezzel a paraméterszám csökkentése és így a tanítási idő illetve erőforrás csökkentése volt. Több érték kipróbálása után a számított futási időt és erőforrást, valamint a loss görbe kezdeti szakaszát figyelembe véve választottam a 8-at megfelelőnek.

Előtanítás Az eredetileg alkalmazott, *dataset*-re vonatkozó paraméterek helyett az általam írt, 3.3.2 fejezetben kifejtett *IterableDataset*-tel dolgoztam. A *batch* méret, azaz az egyszerre feldolgozandó adategységek mérete kétdimenziós: *gradient accumulation* és *gpu* értéket lehet beállítani. A teljes *batch*-méret a kettő szorzata. A tanítás tulajdonságait figyelembe véve döntöttem 32x32 méret mellett, valamint a teszt esetében 32x16 a *batch* mérete. Azok a paraméterek, amelyeket nem említettem, változatlanul maradtak az ajánlotthoz képest. Egy paraméter változtatását fontosnak tartom kiemelni, ez az *mlm probability*, ami a tanulás során a maszkolt tokenek arányát jelenti. A BERT leggyakrabban alkalmazott maszkolási értéke 15%. A DNABERT által ajánlott érték 0.025, azaz 2.5%. A k -merek maszkolása során, $k = 6$ esetén 6 egymás utáni tokent maszkolunk, így a 2.5% maszkolás valójában $6 * 2.5 = 15\%$. A kód nem kezeli ezen maszkolt szekvenciák lehetséges átfedéseit, így az előbbi kiszámolt 15% egy ideális,

maximális érték. A modellt ezért a gyakorlatban érdemes lehet magasabb maszkolási értékkel tanítani. A loss görbe meredek szakaszát figyelve elindítottam néhány tanulást különböző loss értékekkel. Az $mlm = 0.03$ értékkel kaptam optimális eredményeket.

Finomhangolás A finomhangolás paraméterezése az előtanításnál bemutatotthoz hasonló. Az értékeket a *learning rate*-en kívül nem változtattam meg, minden paraméter maradt az eredeti, ajánlott érték.

4.2. Tanítási folyamat

A modell tanításához két darab NVIDIA A100, 40 GB GPU állt rendelkezésre, amelyet a PPKE-ITK HPC klasztere biztosítja.

Előtanítás A modell saját adatokon való előtanítása addig futott, amíg a *loss* értéke nem kezdett el stagnálni. Ez összesen kevesebb, mint egy hétig tartott. Az elfoglalt memória mennyisége a választott paraméterekkel 18000 MiB volt.

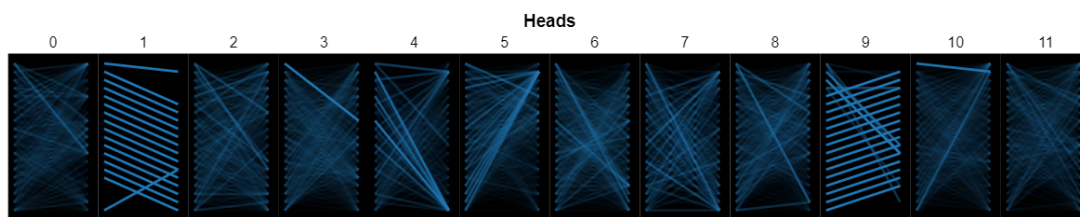
Finomhangolás A finomhangolást két előtanított modellen is elvégeztem. Az egyik az általam, prokarióta mikrobiom adatokon tanított modell, a *ProkBERT*. A másik az eredeti DNA-BERT 6-mereken előtanított változata. A PPD adatbázisból letöltött és feldolgozott adatok halmazán tanítottam. A finomhangolás rövid, paraméterezéstől és az adathalmaz méretétől függően minimum fél óra, legfeljebb 45 perces folyamat GPU-n való tanítás esetén.

5. Eredmények

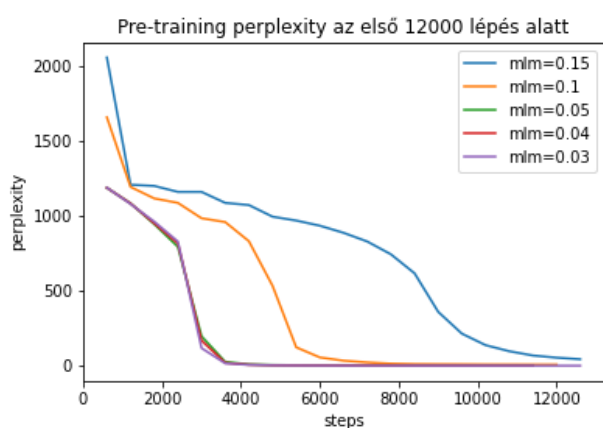
Ebben a fejezetben a modell tanításának eredményei szerepelnek. Először az előtanítás folyamatának eredményeit mutatom be, utána rátérek a finomhangolásról kapott értékekre. Az eredmények kiértékelését a 6. fejezetben végzem.

5.1. Előtanítás

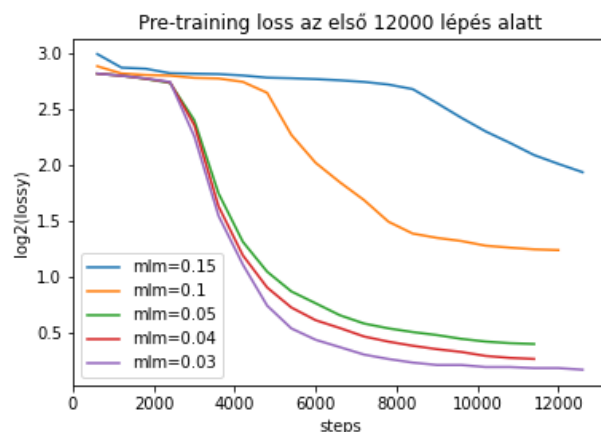
Alább következnek az előtanítás eredményei. Bemutatom az *attention* módszer eredményét, a különböző *mlm* értékkel való kezdeti tanítások összehasonlító ábráját a *loss* és *perplexity* tekintetében, valamint a maszkolási képességet.



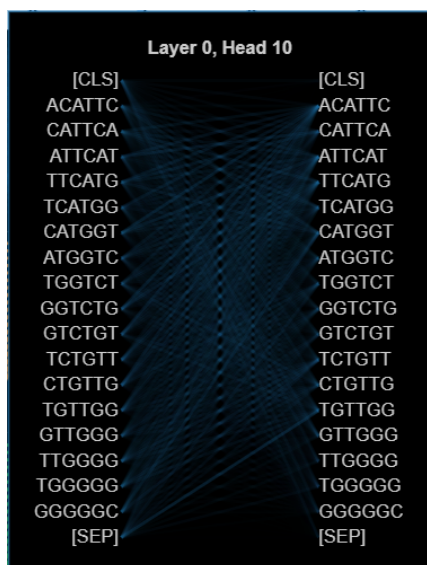
5.1. ábra. Mind a 12 *attention head* által megtanult kapcsolatok, Bertviz[28] használatával ábrázolva.



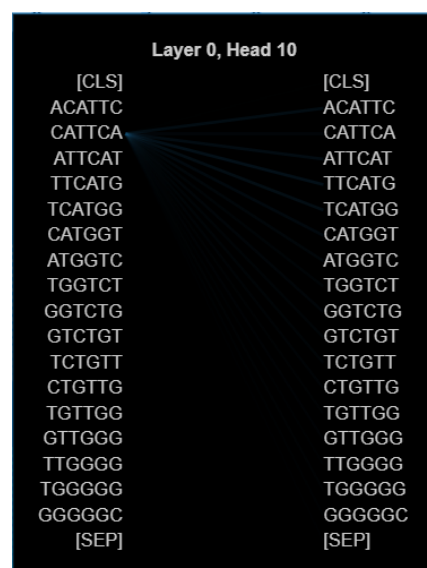
5.2. ábra. Különböző *mlm* értékekkel való előtanítások kezdeti *perplexity* értékei.



5.3. ábra. Különböző *mlm* értékekkel való előtanítások kezdeti $\log_2(loss)$ értékei.



5.4. ábra. Az előtanítás során egyetlen *attention head* által megtanult kapcsolatok a tokenek között. Az ábrázoltást a Bertviz eszköz tette lehetővé.[28]



5.5. ábra. A 12 darab *attention head* célja részben az egymásutáni k-merek közötti kapcsolat felismerése, amely az ábráról leolvasható. Az ábrázoltást a Bertviz eszköz tette lehetővé.[28]

```

examples = [
    "Tomorrow I am going to [MASK] a new BERT model.",
    "GATGCG ATGCGT [MASK] GCGTAA CGTAAT",
    "TGAAAG [MASK] AAAGAG",
    "AATAAA ATGCGT ACTGCT [MASK] TTTTTC AGCAGA"]
for example in examples:
    for prediction in fill_mask(example):
        print(f"{prediction['sequence']}, confidence: {prediction['score']}")
    print("="*50)

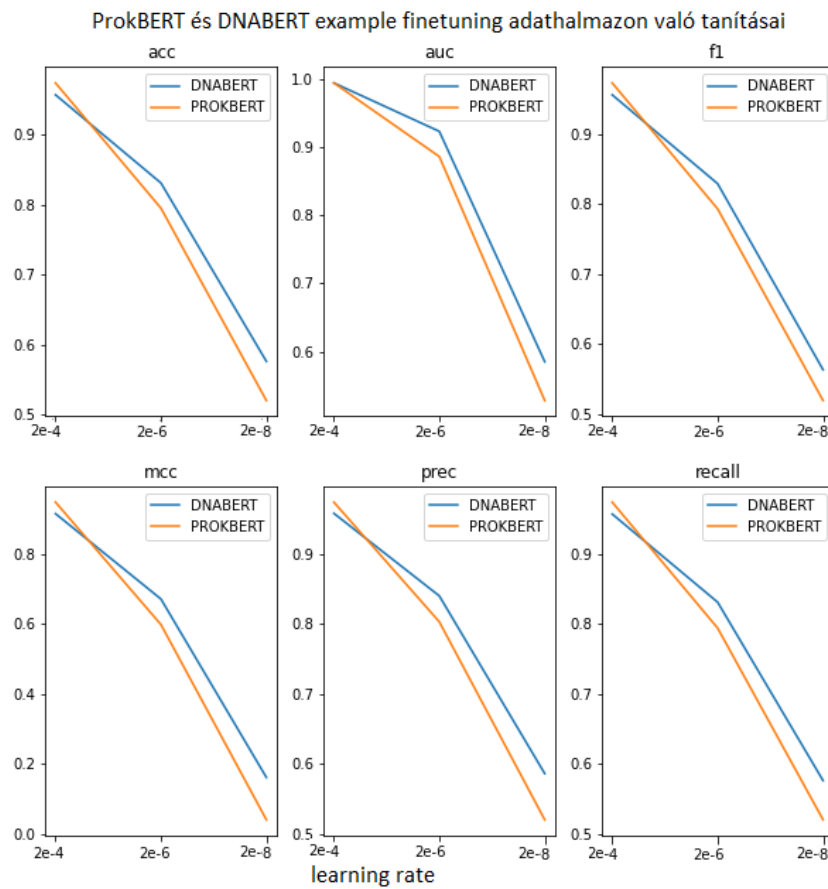
, confidence: 0.9999971389770508
TGATGG, confidence: 8.743882062844932e-09
ATATTA, confidence: 8.459648093150918e-09
ATATTG, confidence: 8.140934149025725e-09
ATATGG, confidence: 7.477012786694104e-09
=====
GATGCG ATGCGT TGCCTA GCGTAA CGTAAT, confidence: 0.9997335076332092
GATGCG ATGCGT TACGTA GCGTAA CGTAAT, confidence: 0.00016574905021116138
GATGCG ATGCGT TGGGTA GCGTAA CGTAAT, confidence: 1.7090735127567314e-05
GATGCG ATGCGT TTCGTA GCGTAA CGTAAT, confidence: 1.3214685168350115e-05
GATGCG ATGCGT TCGCTT GCGTAA CGTAAT, confidence: 1.2205814527987968e-05
=====
TGAAAG GAAAGA AAAGAG, confidence: 0.9865856170654297
TGAAAG AAAAGA AAAGAG, confidence: 0.0043712095357477665
TGAAAG GAAAGC AAAGAG, confidence: 0.004148064646869898
TGAAAG TAAAGA AAAGAG, confidence: 0.0018000999698415399
TGAAAG CAAAGA AAAGAG, confidence: 0.00145292142406106
=====
AATAAA ATGCGT ACTGCT AAAGCA TTTTTC AGCAGA, confidence: 0.4764457643032074
AATAAA ATGCGT ACTGCT AAAATA TTTTTC AGCAGA, confidence: 0.043558329343795776
AATAAA ATGCGT ACTGCT AAAAAA TTTTTC AGCAGA, confidence: 0.033200472593307495
AATAAA ATGCGT ACTGCT AAATAA TTTTTC AGCAGA, confidence: 0.027337195351719856
AATAAA ATGCGT ACTGCT AAAGAA TTTTTC AGCAGA, confidence: 0.023566966876387596

```

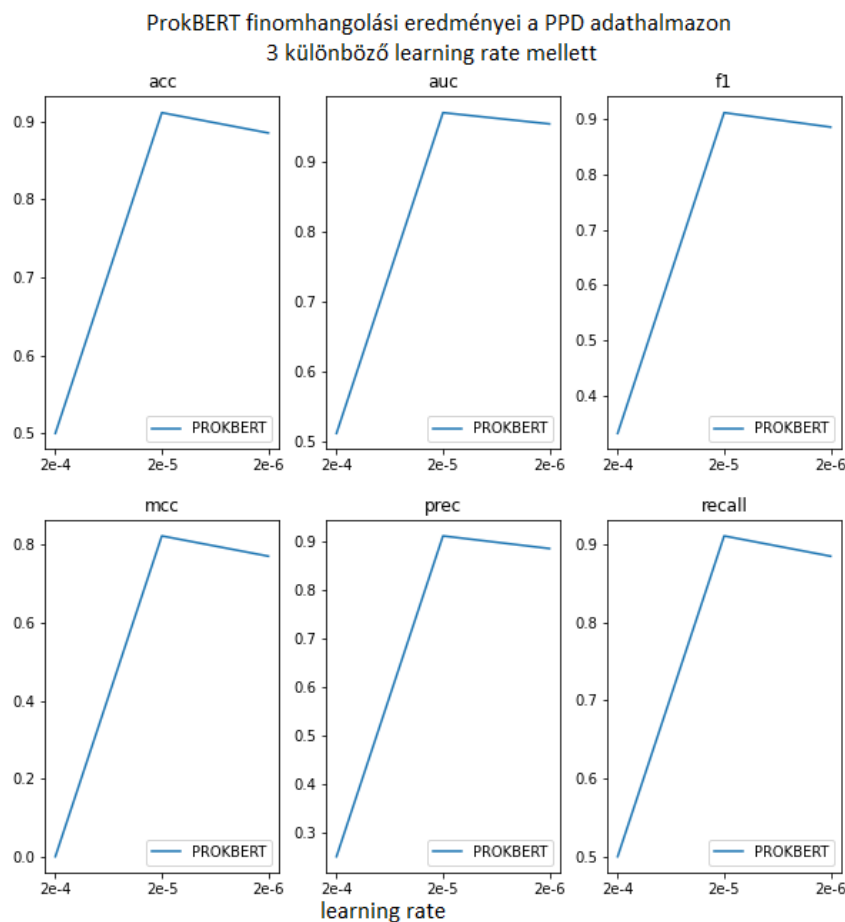
5.6. ábra. Az előtanítás képességének bemutatása szemmel látható módon. A *confidence* értéke mutatja, milyen magabiztossággal prediktálja helyesnek a modell az adott tokenet. Az első mondat egy természetes emberi nyelv mondata. A második és a harmadik egy hosszabb és egy rövidebb DNS szakasz a teszt halmazból. Az utolsó egy random k-mereket tartalmazó szekven-
cia. Látható, hogy a modell a valódi, k-mer formájú szekvenciákkal helyesen prediktál.

5.2. Finomhangolás

A következőkben a különböző paraméterezésű finomhangolások eredményei láthatóak. Az egyes gráfokhoz tartozó alcímek (auc, acc, mcc, stb.) jelentéseit a következő fejezetben, az eredmények kiértékelésekor fejtem ki. Először egy példa-adathalmazon való tanítás során néztem a klasszifikációs feladatra való tanulás képességét. Ezután a *ProkBERT* modellnek a finomhangolási eredményei szerepelnek a PPD prokarióta promótereket tartalmazó adathalmazon való tanítás esetében. Az eredmények kiértékelésekor bemutatott modellekkel 6.3 való összehasonlításhoz elvégzett teszt eredményei a fejezet végén találhatóak.



5.7. ábra. A példa-adathalmazon való finomhangolás eredményei két előtanított modellen, három különböző *learning rate*-tel ábrázolva.



5.8. ábra. A *ProkBERT* finomhangolási eredményei a PPD adathalmazon való tanítás után, három különböző *learning rate*-et vizsgálva.

```
A ppd alapú dataset-en, lr=2e-5 tanított modell értékelési mutatói a ppd dataset-re:
accuracy: 0.912
area under curve: 0.969
precision: 0.914
sensitivity: 0.909
specificity: 0.914
f1 score: 0.911
matthews_corr_coef: 0.823
```

5.9. ábra. A *ProkBERT* $lr=2e-5$ finomhangolási eredményének mérőszámai a PPD teszt adathalmazon nézve.

```
A ppd alapú dataset-en, lr=3e-5 tanított modell értékelési mutatói a ppd dataset-re:
accuracy: 0.912
area under curve: 0.969
precision: 0.912
sensitivity: 0.912
specificity: 0.912
f1 score: 0.912
matthews_corr_coef: 0.824
```

5.10. ábra. A *ProkBERT* $lr=3e-5$ finomhangolási eredményének mérőszámai az PPD teszt adathalmazon nézve.

```
A ppd alapú dataset-en, lr=3e-5 tanított modell értékelési mutatói a benchmarking dataset-re:
accuracy: 0.806
area under curve: 0.906
precision: 0.827
sensitivity: 0.939
specificity: 0.692
f1 score: 0.818
matthews_corr_coef: 0.642
```

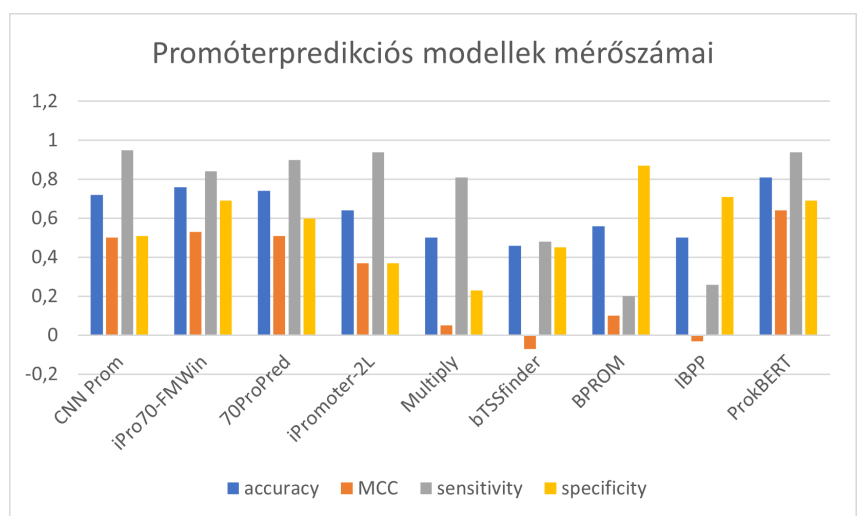
5.11. ábra. A *ProkBERT* lr=3e-5 finomhangolási eredményének mérőszámai az összehasonlító adathalmazon nézve.

```
A ppd alapú dataset-en, lr=2e-5 tanított modell értékelési mutatói a benchmarking dataset-re:
accuracy: 0.802
area under curve: 0.8999
precision: 0.818
sensitivity: 0.923
specificity: 0.697
f1 score: 0.812
matthews_corr_coef: 0.628
```

5.12. ábra. A *ProkBERT* lr=2e-5 finomhangolási eredményének mérőszámai az összehasonlító adathalmazon nézve.

	CNN Prom	iPro70-FM	70ProPred	iPromote	Multiply	bTSSfinder	BPROM	IBPP	ProkBERT
accuracy	0,72	0,76	0,74	0,64	0,5	0,46	0,56	0,5	0,81
MCC	0,5	0,53	0,51	0,37	0,05	-0,07	0,1	-0,03	0,64
sensitivity	0,95	0,84	0,9	0,94	0,81	0,48	0,2	0,26	0,94
specificity	0,51	0,69	0,6	0,37	0,23	0,45	0,87	0,71	0,69

5.13. ábra. Táblázat a különböző promóter-predikciós algoritmusok benchmarking dataset-en való teljesítményének értékelési mutatóival.



5.14. ábra. Összehasonlító táblázat a *ProkBERT* lr=3e-5 tanításának és prokarióta promóter-predikciós modellek ugyanazon adathalmazon való tesztelésének eredményeivel.

6. Eredmények kiértékelése

Ebben a fejezetben a modell tanításának eredményeit elemzem. Először az előtanítás folyamatát nézem meg, az NLP-re jellemző *perplexity* és a *loss* értékeket figyelembe véve. Ezután az előtanítás egyik fő feladatát, a k-mer forma rendszerének felismerését, tehát az egymásutáni szavak közötti kapcsolat meglétét vizsgálom. Majd a finomhangolás eredményeit elemzem. A modellt finomhangolás után egy másik forrásból származó tesztalmazzal is értékelem. Ezen a tesztalmazon kutatók az elmúlt időszak promóter-predikciós algoritmusait vizsgálták és értékelték. Így az eredményeimet más modell ugyanazon tesztalmazon való teljesítményével össze tudom hasonlítani. A modell ezután elhelyezhető az elmúlt évek hasonló algoritmusai között. A fejezetben végül összehasonlítom a munka kezdetén várt és a kapott eredményeket.

6.1. Előtanítás

A *perplexity* a szekvenciákat feldolgozó modellekre jellemző érték. A *perplexity* a legegyszerűbben úgy írható le, mint a prediktált változó lehetséges értékeinek súlyozott összege. Ezt a mennyiséget matematikailag a *shannon entropy*, azaz a jelenlévő bizonytalanságot leíró formula alapján lehet levezetni. Minél kisebb ez az érték, annál jobb a modell teljesítménye. Más modellekkel való összehasonlításra nem alkalmas a *perplexity*, mivel ez az inputoktól, a prediktált tokenektől függ. Ugyanazon a tanító és tesztalmazon való tanítás esetén különböző konfigurációjú modellek teljesítményének vizsgálatára azonban megfelelő támaszt nyújt. Éppen ezért a megfelelő konfiguráció és paraméterezés kiválasztásához ezt az értéket is figyelembe vettem.

A modell tanításának kezdetén a *perplexity* nagy. A nem megfelelő beállításokkal 4000 – 2000, a tanításhoz kiválasztott végső értékekkel 1200 volt a *perplexity* kapott értéke az első *evaluation* során. Ez gyorsan csökkent az első párezer *batch* lefutása során. A második 12 órás futásra a *perplexity* 1000, 900 majd 120 és 20 alá csökkent. Ezeket a korai értékeket vettem kritikus értékeknek a megfelelő beállítások kiválasztása közben. Megfigyeléseim szerint ugyanis az első egy-két nap futás eredménye alapján következtetni lehet a *perplexity* csökkenésének további mértékére. 5.2

A *loss* érték fontos támpont minden gépi tanulási módszer teljesítményének méréséhez. Különböző *loss*-t számító függvények léteznek a modell fajtájától függően, például *mean squa-*

red error a regressziós vagy *cross-entropy* a klasszifikációs problémákhoz. A BERT modell az utóbbit használja a tanulás során.

A tanult súlymátrixok (lásd 2.2. fejezet) értékeinek frissítéséhez a modell minden *batch* után kiszámolja a prediktált tokenek hibáját. A hiba mértékének kiszámítására a következő az egyenlet:

$$e = \sum_{i=1}^n -\tilde{Y}_i \log(Y_i) \quad (6.1)$$

Ahol $\tilde{Y}$ a várt érték, Y a prediktált, n pedig az összes predikció amellyel számol a modell. Az eredmény alapján ezután frissíti a súlyokat. Ezek következő értékét az előző súly, a *learning rate* és a hiba grádiense alapján kapja meg.

A *loss* görbe a *perplexity* 1200-as értéke mellett 7-8 körüli értékről indul. Kezdetben a *perplexity* változásához hasonlóan gyorsan, majd egyre kevésbé csökken az értéke. A különböző beállítású tanítások *loss* görbáját a kettes alapú logaritmus értékek alapján ábrázoltam az első 12000 lépés alatt. 5.3

Az előtanítás során a modell félig-felügyelt tanulással a szekvenciákban belüli kapcsolatokat tanulja meg. Minél több *attention head*-et használunk, annál több szempontból figyel az összefüggésekre. A DNS nyelv ismeretlen voltából adódóan a számunkra könnyen ellenőrizhető kapcsolat csak a k -merek közötti, tehát a közvetlen egymás után következő tokenek kapcsolata. A Bertviz eszköz [28] lehetővé teszi az egyes fejek által vizsgált kapcsolatok ábrázolását a tokenek szintjén. A fejek között található olyan, amely a k -merek közötti összefüggést mutathatják.

5.5 5.4 5.1

Az előtanított modell teljesítményét végül számítások nélkül, a predikciós képesség "kipróbálásával" is megnéztem. Ugyan ez a módszer nem ad megbízható és összehasonlítható értékelést, a *perplexity* és *loss* értékek mellett mégis egyszerűen látható, mire képes a modell.

Ehhez a vizsgálatához jupyter notebookban a betöltött modellnek különféle mondatokat adtam: a természetes emberi nyelv egy mondatát, két átfedő k -mer formájú szekvenciát és egy random k -merekből álló szekvenciát, mindből egy-egy tokenet maszkolva. A modell a hiányzó tokenet prediktálja, valamint egy magabiztossági értéket is megad hozzá. Így könnyen belátható, hogy az átfedő k -mer alapú szekvenciákban valóban nagy valószínűséggel találja el a hiányzó tokenet. Az első prediktálandó mondattal látható, hogy a szótárban nem szereplő tokeneket használva, például emberi nyelv mondataira üres tokenet ad vissza nagy magabiztossággal. Az utolsó sorban random k -merek szekvenciájára ellenőriztem a modellt, erre kicsi magabiztossággal tippel tokeneket. 5.6

6.2. Finomhangolás

A finomhangolás folyamatát két előtanított modellen végeztem többféle paraméterezéssel, a DNABERT által közzétett példa-adathalmazon, illetve a PPD-ről letöltött promótereket tartalmazó adathalmazon tanítva, ahogy a 4.2 részben olvasható. A modellek teljesítményének kiértékelését a BERT modell tanításához tartozó értékelési forma alapján végeztem, azaz hat mérőszámot vettem figyelembe. Ezek a mérőszámok segítenek látni a finomhangolt modellben rejlő potenciált, azonban a végső célra való alkalmazhatóságukról kevésbé adnak információt. Csakúgy, mint a DNABERT, az általam tanított modell célja is szekvenciák elemzése, egyes régiók felismerése. A következőkben bemutatott kiértékelés abban az esetben adja meg a modell klasszifikációs képességeit, amennyiben rövid szekvencia-darabok egészét szeretnénk osztályozni. Ezekkel az értékekkel más klasszifikációs modellekhez lehet hasonlítani a modell képességeit. Amennyiben a modell megfelelően teljesít ezen osztályozási feladatok kapcsán, úgy érdemes megvizsgálni a hosszú szekvenciák elemzésének feladatát, amely egy ún. ablakolós módszerrel történik.

6.2.1. Kiértékelési mutatók

A következőkben bemutatom az értékeléshez használt mutatókat. A modell bináris klasszifikációs feladatra lett finomhangolva, ennek megfelelően az értékelési mutatókat kétosztályos problémákra vonatkozóan mutatom be. Megértésükhöz először fontos megismerni a *confusion matrix*-ot.

Négy kategóriába sorolhatóak a prediktálás eredményei. A pozitív minták között, a helyesen prediktált a True Positive (TP), míg a negatívnak címkézett minta a False Negative (NF). True Negative (TN) az a negatív minta, amelyik negatívnak is lett prediktálva, míg a helytelenül pozitívak közé soroltak a False Positive (FP) esetek. A következőkben bemutatott értékelési mutatók ezen számok különböző arányai alapján kaphatóak meg.

ACC - accuracy Az első érték, amelyet bemutatok, az *accuracy*. Klasszifikációs modellek esetében ezt az értéket a helyesen prediktált minták és az összes eset arányaként kapjuk meg. Bináris klasszifikációs problémákra és többosztályos feladatok értékelésére is alkalmazható. Az *accuracy* nagy hátránya, hogy kiegyensúlyozatlan adathalmaz esetében, amikor aránytalanul nagy mennyiségű minta van jelen az egyik osztályból, nem mindig ad megbízható képet a

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

6.1. ábra. Confusion matrix. Forrása: <https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning>[29].

teljesítményről.[30][31] A jelenlegi tanítás esetében nagyon kevés pozitív minta áll rendelkezésünkre, amelyhez képest rengeteg a generált negatív minta. Így az *accuracy* értékét óvatosan kell kezelni, nem feltétlen ad értékes információt.

AUC - area under curve A következő az *AUC - Area Under Curve*, azaz görbe alatti terület. Elsődlegesen bináris klasszifikációs feladatokra alkalmazott érték.[29] Az igazi pozitív és összes pozitív minta aránya a később szereplő *recall*. A *recall* és a hamis pozitív/összes negatív aránynak egymáshoz viszonyított értékéből kapjuk a *receiver operator characteristic*, *ROC* görbét, melyet ábrázolva, a görbe alatti terület mértéke ad információt a modell klasszifikációs képességéről.

MCC - Matthews correlation coefficient A *Matthews correlation coefficient* a bináris osztályozási feladatok értékelésének egy mérőszáma. Egyes kutatók véleménye szerint megoldás lehet az egyenetlen dataset problémára is. [32] Az MCC a helyes predikciók korrelációját méri az összes címkézéshez, értéke -1 és 1 közötti lehet.

F1 Az *f1 score* az MCC-hez hasonló fontosságú érték. True negatívától független és osztályra van definiálva, így az f1 pont főleg akkor fontos, ha a két osztály nem egyenlő mértékben számít.

A definiálásból következik, hogy a pozitív és negatív minták címkéit felcserélve megváltozik az értéke.

Precision A *precision* érte a helyesen prediktált pozitív minták aránya az összes pozitívnak prediktált között. A negatív minták egyáltalán nem szerepelnek ennek az értéknek a definíciójában, így az eredmény, főleg egyetlen adathalmaz esetén, nem megbízható.

Recall A *recall* a jól prediktált pozitív minták az összes valódi pozitív számához viszonyítva.

6.2.2. Tanítások eredményei

Most következik a finomhangolás eredményeinek elemzése.

Finetuning example adatok Elsőként a DNABERT által feltöltött *finetuning example* adatokon finomhangoltam mind a DNABERT $k = 6$ -mer, mind a *ProkBERT* előtanított modelleket. Három különböző *learning rate*-tel indítottam a tanításokat, így hasonlítottam össze a kettő modell klasszifikációs feladatra való képességét. A 5.7 képen jól látható, hogy az általam előtanított modell eredményei a DNABERT előtanított modellhez hasonlóan alakulnak. Mivel ez csak egy példa adathalmaz, a pontos eredmények és eltérések nem fontosak számomra. Ennek a feladatnak a célja az, hogy az előtanított modellt lehessen ellenőrizni. Így látható, hogy a finomhangolást a valódi tanítóhalmazon is érdemes megkezdni.

PPD adathalmaz Miután sikeresnek ítéltam a *ProkBERT* előtanítását, elkezdtem a PPD-ről letöltött promótereket tartalmazó adathalmazzal való finomhangolást. Először három különböző *learning rate*-tel tanítottam, majd ezek értékelési mutatóit hasonlítottam össze. A 5.8 ábra mutatja az eredményeket. Ezek alapján a $2e - 5$ nagyságrendű *learning rate*-et gondoltam a legalkalmasabbnak, így ekörüli értékekkel próbálkoztam tovább.

A további finomhangolások közül végül a $lr = 2e - 5$ és a $lr = 3e - 5$ rendelkezett a legmagasabb értékelési mérőszámokkal. Mindkét eredmény esetében magas a helyesen prediktált értékek, míg alacsony a fals predikció száma.

Ennek a két modellnek a korábban bemutatott értékelési mérőszámait a 5.9 és a 5.10 ábrákról lehet leolvasni. Mivel a tanító halmaz egyenlő arányban tartalmazott valódi promóter szekvenciákat és negatív mintákat, a mérőszámok valós képet adnak a teljesítményről. Mindkét modell nagyon hasonlóan kis mértékben téved a pozitív és a negatív minták esetében. Ezek

a magas értékek azt mutatják, a *ProkBERT* modell a PPD adathalmazon való tanulás esetén szintén PPD-ről származó teszhalmazon kiválóan teljesít.

6.3. Validáció benchmarking adathalmazon

A finomhangolt *ProkBERT* eredményét végül összehasonlítottam több promóterpredikciós célra alkalmazott algoritmussal. Ehhez egy 2020-as cikket vettem segítségül, a *Benchmarking Bacterial Promoter Prediction Tools: Potentialities and Limitations* című publikációt.[26] Ebben nyolc, baktériumpromótereket prediktáló algoritmust hasonlítottak össze a szerzők. Az összehasonlításhoz a regulonDB-ről letöltött promóterszekvenciák és a generált negatív minták keverékét használták teszhalmazként. A *ProkBERT* finomhangolásához való tanító halmaz összeállításakor ügyeltem arra, hogy ne kerüljenek bele a regulonDB promóterei között is szereplő szekvenciák. Így anélkül tesztelhettem az adathalmazon a modellt, hogy az a tanítás során a teszhalmazban szereplő példákra tanult volna.

A cikk írói egy táblázatba foglalták a vizsgált predikciós algoritmusok tulajdonságait. 5.13 A különböző modellek között volt neurális háló, döntési fa és HMM modell is. A kutatók feltüntették a tanító adathalmazt is, valamint azt, hogy az algoritmusok hány különböző *E. coli* szigma-faktoron tanultak. Több szempontból értékelték a modelleket, jelen kutatásomhoz azonban csak a validációs halmazon való tesztelés értékei szükségesek. A *ProkBERT*-tel való munka ugyanis még kezdeti szakaszában tart, így az alkalmazhatóságára, felhasználói felületére vonatkozó összehasonlításokat korai lenne végezni. A cikk írói egy oszlopdiagramon ábrázolták a tesztelés eredményeit az egyes predikciós eszközökre nézve. Négy értékelési mérőszámot néztek: *accuracy*, *MCC*, *sensitivity* és *specificity*. A grafikonon látható, hogy a nyolc modell fele egyértelműen magasabb értékekkel rendelkezik, míg a másik fele erősen lemaradó értékeket mutat. A legszembevetőbb különbséget az *MCC* értéke mutatja, amely nulla közeli, akár negatív értékeket is felvesz a négy lemaradó modell esetében. Általánosan az $MCC = 0$ érték a random predikció teljesítményével egyezik meg. Ahhoz, hogy a *ProkBERT* teljesítményét könnyedén össze tudjam hasonlítani ezekkel az eredményekkel, a kutatásban szereplő ábrához hasonló módon, oszlopdiagramokkal mutatom be a nyolc modell, valamint mellettük szerepelve a *ProkBERT* mérőszámait.5.14 5.12 5.11 Az adatok alapján elmondható, hogy a *ProkBERT* a négy jobban teljesítő algoritmus közé sorolható. Az eredmények pontosabb meghatározásához azonban a *ProkBERT* további teszteléseire, validációjára van szükség, anélkül további

megállapításokat nem lehet tenni.

6.4. Konklúzió

A *ProkBERT* modellel való munka egy érdekes tanulmányként indult a transzformer alapú nyelvmodellek prokarióta DNS-szekvencián való tanításának vizsgálatára. A *DNABERT* modell a humán genomon való tanítás után bizonyította, hogy ezen gépi tanulós algoritmusok alkalmasak az eukarióta genom analízisére.[19] A kutatás publikálása óta ugyan többen sikeresen finomhangolták baktérium genomon az előtanított DNABERT modellt, prokarióta genomon való előtanításról azonban nem szóltak kutatások. A *ProkBERT* eredményei mind az előtanítás, mind a finomhangolás után ígéretesek. A modellben rejlő potenciált további kutatásokkal szükséges vizsgálni, például egyéb osztályozási feladatok vizsgálatával és validációs lépésekkel. A szekvencia egészének analízisére irányuló képességét kutatásom során nem vizsgáltam, azonban ez fontos, hogy jobb képet kapjunk a modell teljesítményéről. Ehhez a szekvenciák ablakolás módszerrel való vizsgálatát kell elvégezni és értékelni. Bár ezen további feladatok elvégzését fontosnak tartom a modell teljeskörű értékeléséhez, jelen kutatásom eredményei alapján kijelenthetem, hogy a *ProkBERT*-tel való munka sikeres, a jövőben érdemes tovább foglalkozni vele.

7. Összefoglalás

Ebben a fejezetben összefoglalom az elvégzett munkám és annak eredményeit. Röviden leírom a következőkben elvégzendő további feladatokat. Végül az eddigi eredmények tekintetében a távolabbi jövőre való kitekintésről lelsz szó.

7.1. Feladatok

Munkám során egy transzformer alapú természetesnyelv-feldolgozó modellel foglalkoztam. Célom annak felderítése volt, hogy egy ilyen modell alkalmas-e meglévő prokarióta-genom tudásbázisunkat hasznosítani, amennyiben az ismert, címkézett adat kevés. Ehhez először a humán bél-mikrobiom adatokat kellett megfelelő módon feldolgoznom. Majd a kiválasztott BERT-modellt módosítottam úgy, hogy ezen DNS-szekvenciák feldolgozására legyen képes. Ezután az előfeldolgozott DNS adathalmazon előtanítottam, majd prokarióta promótereken finomhangoltam a modellt. A finomhangolásra kiválasztott feladat a prokarióta promóterek predikciója, egy kétosztályos klasszifikációs feladat. A modell végül a *ProkBERT* nevet kapta.

7.2. Eredmények

A *ProkBERT* modell osztályozási képességét teszteltem a PPD-Prokaryotic Promoter Database-ről letöltött adatokból előállított teszthalmazon. A modell ezen a teszthalmazon kiválóan teljesít, magas értékelési mérőszámokkal, mint például az AUC, accuracy vagy MCC. Az utóbbi években alkalmazott prokarióta promóter-predikciós eszközöket egy 2020-ban megjelent cikk összehasonlította, amely eredmények segítették a munkám értékelését. A vizsgált algoritmusokkal való összehasonlíthatóság érdekében a finomhangolt *ProkBERT* modellt a cikk írói által előállított, regulonDB-ről származó teszthalmazon is kiértékeltem. Az eredmények alapján a *ProkBERT* promóter-predikciós klasszifikációs feladatra tanított változata felveheti a versenyt a cikk szerzői által legjobb négynek ítélt predikciós algoritmusokkal.

A modell teljesítményét ezenfelül más adathalmazokon, illetve más módszerekkel is szükséges validálni, amennyiben magabiztosabb kijelentéseket szeretnénk tenni az értékelési mérőszámokkal kapcsolatban.

7.3. Jövőbeli tervek

Munkám a modell tanításával és értékelésével nem ér véget. Az eredmények arra engednek következtetni, hogy a kutatás jó irányba halad, érdemes tovább foglalkozni a transzformer alapú nyelvmodellek prokarióta DNS-szekvencia elemzésére való alkalmazásával.

A *ProkBERT* modellt egyelőre egyetlen osztályozási feladatra való tanítással vizsgáltam. A továbbiakban a definiált osztályozási feladatok körét lehet bővíteni: gén-predikció, génklaszter címkézés. Ezen feladatokra való tanítás teljesítményének kiértékelése segíti a modell elhelyezését hasonló algoritmusok között.

A klasszifikációs problémákon felül a modell képességeit a jövőben az ablakolós módszerrel is érdemes megvizsgálni. Így arról kapnánk képet, hogy egy teljes szekvenciát kapva inputként, mennyire pontosan képes a modell a promóter szakaszokat felismerni és pozicionálni.

A feladatok esetleges limitációi eredhetnek az előtanítás adathalmazának szűk voltából, ez ugyanis a mikrobiális közösségeknek csak kis része, a humán bél-mikrobiom. A jövőben ezért érdemes lehet az előtanító korpusz méretét növelni, majd a finomhangolási feladatokat az új modellel értékelni és validálni.

A *ProkBERT* sikeres tanítása és validációja hatalmas előrelépést jelentene mind a mikrobiom vizsgálatára irányuló kutatások, mind az orvostudomány terén. A jövőben a modell segítségével hatékonyan lehetne felhasználni a rendelkezésre álló, akár csak kevés mintából álló adatokat is. Így végül a különböző feladatokra finomhangolt *ProkBERT* akár a személyre szabott mikrobiom terápia alkalmazásához járulhatna hozzá.

Irodalom

- [1] E. R. W. Barbara J. Culliton, „GNN, Genomes News Network“, 2000.
- [2] M. Gu‘vi, „THE HISTORY OF DNA SEQUENCING, ISTORIJAT SEKVENCIRANJA DNK“, 2012.
- [3] B. C. James M. Heather, „The sequence of sequencers: The history of sequencing DNA“, 2015.
- [4] . R. K. K. Aditya H, „Bioinformatics: Bioinformatics and biological data mining“, 2022.
- [5] L. Stein, „Genome annotation: from sequence to biology“, 2001.
- [6] A. S. Proc Natl, „Distinguishing protein coding and non coding genes in human genome 104(49):19428-33.“, 2007.
- [7] S. Aryal, „The Biology Notes: Gene Prediction- Importance and Methods“, 2022.
- [8] Y. L. Zhuo Wang Yazhu Chen, „A Brief Review of Computational Gene Prediction Methods“, 2004.
- [9] e. a. Aimin Yang, „Review on the Application of Machine Learning Algorithms in the Sequence Data Mining of DNA“, 2020.
- [10] N. et al, „DNA Sequence Classification by Convolutional Neural Network. Journal of Biomedical Science and Engineering, 9, 280-286. doi: 10.4236/jbise.2016.95021.“, 2016.
- [11] D. G. M. Bosco GL, „Deep learning architectures for DNA sequence classification, International workshop on fuzzy logic and applications, Springer, pp 162–171“, 2016.
- [12] K. D., „A Brief History of Natural Language Processing (NLP)“, 2019.
- [13] e. a. TIANYANG LIN YUXIN WANG, „A Survery of Transformers“, 2017.
- [14] e. a. Salman Khan Muzammal Naseer, „Transformers in Vision: A Survey“, 2017.
- [15] N. P. Ashish Vaswani Noam Shazeer, „Attention Is All You Need“, 2017.
- [16] M. Saeed, „A Gentle Introduction to Positional Encoding In Transformer Models“, 2022.
- [17] e. a. Feiqi Cao Soyeon Caren Han, „Understanding Attention for Vision-and-Language Tasks“, 2022.

- [18] J. D. M.-W. C. K. L. K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“, 2019.
- [19] Z. Z. et al Yanrong Ji, „Pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome“, 2021.
- [20] J. DEKKER, „Gene Regulation in the Third Dimension“, 2008.
- [21] e. a. Yifeng Li Chih-yu Chen, „The identification of cis-regulatory elements: A review from a machine learning perspective“, 2015.
- [22] J. G. Nina Shenker-Tauris, „IMPROVED METAGENOMIC BINNING WITH TRANSFORMERS“, 2022.
- [23] E. A. C. James Chen Hande Boyaci, „Diverse and unified mechanisms of transcription initiation in bacteria“, 2021.
- [24] R. R. Harley C.B., „analysis of E.coli promoter sequences. Nucleic Acids Res.“, 1987.
- [25] A. B. Chowdhury Nilkanta, „Comparative analysis of prokaryotic and eukaryotic transcription factors using machine-learning techniques“, 2018.
- [26] R. S.-R. Murilo Henrique Anzolini Cassiano, „Benchmarking Bacterial Promoter Prediction Tools: Potentialities and Limitations“, 2020.
- [27] S. Labs, „Language models: battle of the parameters“, 2020.
- [28] J. Vig, „A Multiscale Visualization of Attention in the Transformer Model“, 2019.
- [29] A. Bhandari, „AUC-ROC Curve in Machine Learning Clearly Explained“, 2020.
- [30] S. S. Sokolova M Japkowicz N, „Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation“, 2006.
- [31] C. Z. Gu Q Zhu L, „Evaluation measures of the classification performance of imbalanced data sets“, 2009.
- [32] G. J. Davide Chicco, „The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation“, 2020.