#### УНИВЕРЗИТЕТ УНИОН

#### РАЧУНАРСКИ ФАКУЛТЕТ

Од слова до речи,

oд биграма до ГПТ-а:

Велики језички модели

#### Дипломски рад

Ментор: Кандидат:

Немања Илић Михајло Маџаревић

Београд, 2024.

Садржај

[1. Увод 3](#_Toc177494608)

[2. Обрада природних језика 4](#_Toc177494609)

[2.1. Рачунарска лингвистика 4](#_Toc177494610)

[2.2. Велики језички модели 4](#_Toc177494611)

[2.2.1. Кратка историја великих језичких модела 5](#_Toc177494612)

[2.3. Кораци обраде природног језика 6](#_Toc177494613)

[2.3.1. Обрада и чишћење података 6](#_Toc177494614)

[2.3.2. Дефинисање архитектуре модела 9](#_Toc177494615)

[2.3.3. Тренинг 10](#_Toc177494616)

[2.3.4. Евалуација 13](#_Toc177494617)

[3. Имплементација 14](#_Toc177494618)

[3.1. Чишћење и обрада података 14](#_Toc177494619)

[3.2. Биграм модел 15](#_Toc177494620)

[3.2.1. Софтмакс активациона функција 15](#_Toc177494621)

[3.3. ГПТ модел 17](#_Toc177494622)

[3.3.1. Трансформер глава 18](#_Toc177494623)

[3.3.2. Пажња више трансформер глава 20](#_Toc177494624)

[3.3.3. Слој пропагације унапред 21](#_Toc177494625)

[3.3.4. Трансформер блок 23](#_Toc177494626)

[3.3.5. Функција пропагације унапред 25](#_Toc177494627)

[4. Резултати и дискусија 26](#_Toc177494628)

[4.1. Биграм модел слова 26](#_Toc177494629)

[4.2. Биграм модел речи 28](#_Toc177494630)

[4.3. Трансформер модел слова 29](#_Toc177494631)

[4.4. Трансформер модел речи 31](#_Toc177494632)

[5. Закључак 34](#_Toc177494633)

[6. Литература 35](#_Toc177494634)

# 1. Увод

Последњих година језички модели су заинтересовали јавност. Велику пажњу су привукли појавом нових генеративних претренираних трансформера (ГПТ - а). Могућност пружања одговора природним људским језиком на питања било које тематике на било ком језику је изненађујућа. Њихова моћ се огледа у превођењу, пружању одговора у реалном времену, издвајању специфичних тражених података, класификацији, анализи и још много делатности. Велики језички модели се користе као претраживачи, за преуређивање текста и проналажење језичких грешака, у решавању проблема природних наука, у програмирању [14]. Још један занимљив начин примене модела је промпт инжењерство. Промпт подразумева било какав текст који ми као корисник проследимо великом језичком моделу. Идеја промпт инжењерства је да ми као корисник моделу задамо инструкције понашања које ће он пратити [17]. На овакав начин можемо направити чет бота за пицерију, продавницу, асистента на веб страници и слично. Одавде се већ може приметити велика корист и доступност ових модела на тржишту.

У овом раду је показан поступак настанка великог језичког модела од самог почетка, једноставније као и сложеније архитектуре попут генеративног претренираног трансформера. Показан је поступак обраде и припреме текста за тренинг модела, разлике у перформансама модела приликом одабира различитих типова “токена” за тренинг као и разлике у перформансама модела изменом подесивих параметара модела.

# 2. Обрада природних језика

Обрада природних језика је вид машинског учења које рачунарима пружа могућност превођења, контроле и разумевања људског језика [13]. Велике организације прикупљају огромне количине података чија се обрада може аутоматизовати употребом језичких модела. Како дакле обрађивање природних језика ради? У наставку ћемо се упознати са поступцима обраде и припреме текста, примене модела у обради ових података као и самог тренинга.

## 2.1. Рачунарска лингвистика

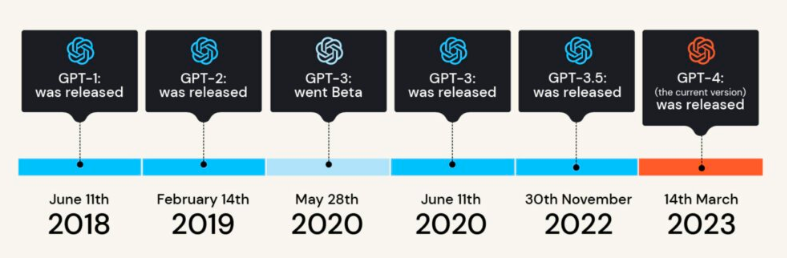
Рачунарска лингвистика је наука која изучава конструисање језичких модела вештачке интелигенције. Научници се баве синтаксном и семантичком анализом у покушају да машинским моделима приближе стандардни људски језик. Софтверски преводиоци, текст-говор синтисајзери и препознаваоци говора су установљени на рачунарској лингвистици [15]. Да би се резултати добијени рачунарском лингвистиком употребили користи се машинско учење као и дубоко учење.

## 2.2. Велики језички модели

Велики језички модел представља алгоритам за дубоко учење који користи огроман број параметара и података за тренинг да би разумео и предвидео текст. Ови типови модела вештачке интелигенције су способни да генеришу, ревидирају и преводе текст. Реч велики из назива ових модела се односи на број параметара, променљивих и тежина који ови модели користе. Примера ради Гугл - ов БЕРТ модел има 340 билијарди параметара, док ОпенАИ - ов ГПТ четири модел има 1,8 трилијарди параметара [17]. Ови параметри су подесиви бројеви и модел их учи током тренинга (мења вредност самог броја како би прилагодио да дати улаз даје очекиван излаз модела).

### 2.2.1. Кратка историја великих језичких модела

Од 1906. до 1912. године Фердинанд де Сосир је изучавао Индоевропску лингвистику и генералну лингвистику на Женевском универзитету. Током овог периода поставио је веома функционалан модел језика као система. Умро је 1913. без објављивања сопственог рада, али су његове колеге Алберт Сешеје и Шарл Бали сачували његов рад и потрудили се да направе књигу Кур де лингвистик (преведено Курс лингвистике), што се преточило у процесирање природних језика објављено 1916. Крајем рата 1945. поље обраде природних језика придобило је велику пажњу. Мировни разговори, жеља за интернационалном трговином и потреба међусобног разумевања су поспешивали наду прављења машине која би могла да преводи језике. Посао прављења преводиоца није нарочито једноставан, људски језик је пун нелогичних правила и хаотичности, али математички језик није. Артур Самуел, тада запослен у фирми ИБМ, направио је програм за играње игре Дама и за овај програм је направио низ алгоритама који би наводили програм да се побољша у игри за које је дат опис да машина учи. Ово је уједно и први пут употребе термина машинског учења 1959. године [16]. Јосеф Вајзенбаум је 1966. направио ЕЛИЗУ, што се сматра првим програмом обраде природног језика. ЕЛИЗА је могла да схвати улаз и одговори са предетерминисаним одговором. Није користила базу података већ се ослањала на преуређивање улаза корисника [17]. Машинско учење је дуго време стагнирало због техничких ограничења попут мало меморијског простора, недовољно података, споре брзине процесирања. Машинско учење се углавном ослањало на вероватноћу и статистику. Први развој језичког модела је почео ИБМ 1980-их чији је задатак био да предвиди следећу реч у реченици. Модел се ослањао на речник који је имао вероватноћу појаве сваке речи. После сваке речи алгоритам је израчунавао нове статистике за речи. Тим Бернер Ли-ова замисао светски доступног интернета је постала реалност 1991. и пружила велике могућности прикупљања података за језичке моделе. Настанак дубоког учења 1990-их је био значајан за развој језичких модела, мада је тек 2011. године област почела придобијати популарност [16]. Први велики језички модели су настали 2017. појавом рада о трансформерима [6], новој архитектури дубоке неуронске мреже. Модели попут Гугл – овог БЕРТ – а и ОпенАИ – овог ГПТ – а. На Слици 1 можемо видети историју развоја различитих верзија ОпенАИ ГПТ модела, док на Слици 2 можемо видети развоје у индустрији машинског учења последњих десет година које су највише утицале у настанку модела за које знамо данас.



Слика 1. Историја ГПТ модела (извор: [1])



Слика 2. Историја језичких модела последњих десет година (извор: [2])

## 2.3. Кораци обраде природног језика

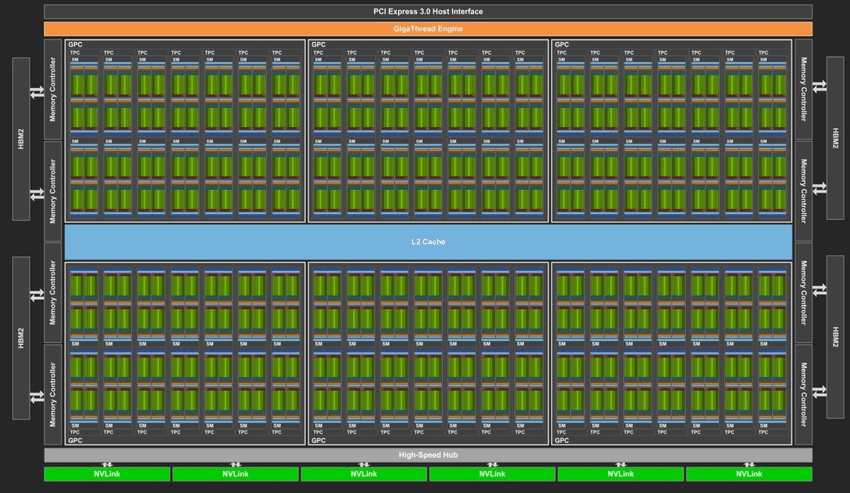
Имплементација уобичајено подразумева прикупљање неструктурираног текста, података говора са разних извора попуд клауд података, имејлова, пословних апликација, интернет чланака и слично, затим дефинисање архитектуре модела па тренинг и на крају евалуација самог модела.

### 2.3.1. Обрада и чишћење података

У овом кораку се улазни подаци чисте од нежељених симбола или речи и припремају за обраду модела. Речи се могу свести на свој корен ради бољих резултата модела и оне које нису од велике важности се могу уклонити, потом се улаз енкодира у ”токене”. Постоје пар различитих могућности приликом одабира облика типа “токена”. “Токени” могу бити слова, слогови или пак читаве речи. “Токени” се затим претварају у бројеве. Овај процес се назива енкодирање. Енкодирање је неопходно јер рачунари не разумеју текстуалне податке. Улаз модела мора бити у формату броја. Посао енкодирања уобичајено ради енкодер. Поред енкодера постоји и декодер који ради обрнути посао дешифровања секвенце бројева у “токене” који идеалистично представљају смислен текст. Декодери су неопходни како бисмо секвенцу бројева коју модел даје као излаз претворили у текст који људи могу прочитати.

Подаци се деле на тренинг и валидациони скуп, поред ова два скупа може се увести и тест скуп за бољу процену квалитета модела. Тренинг подаци се користе за сам тренинг модела о којем ће се више говорити у наставку рада. Валидациони скуп учествује у тренингу, спречава модел да напамет научи тренинг податке, тачније обезбеђује да модел може добро да генерализује, даје процену колико је наш модел добар у изради задатка за који је намењен. Тест скуп служи крајњој процени модела после свих тренинга. Тест скуп су подаци које модел никада није видео током тренинга и служе као добар показатељ прецизности нашег модела. Поред ових скупова подаци се могу делити на серије које се још називају “бечеви”. Ова подела се првенствено врши зарад бржег тренинга, али има и бројне друге бенефите попут боље генерализације модела, брже конвергенције модела, мањег рачунарског оптерећења и слично.

Све ове погодности су присутне уколико користимо графичку карту за тренинг модела. Оне су погодније за тренинг јер садрже велик број аритметичко логичких јединица (АЛУ) које су задужене за рачунске операције. Процесори имају једну или неколико комада аритметичко логичких јединица, али свакако много мање неко графичке карте. На Слици 3 је приказана Нвидиа V100 графичка карта.



Слика 3. Приказ количине АЛУ (зелени правоугаоници) јединица Нвидиа графичке карте (извор: [3])

Сама конкуретност која се дешава је појава која настаје због великог броја АЛУ јединица у графичким картама, тачније можемо разумети да се “беч” дели на више АЛУ јединица. Одабир величине “беча” има утицај на тренинг па самим тиме и на перформансе модела. Просто речено “беч” можемо замислити као засебни скуп примера који моделу дајемо ради тренинга. Оваквих скупова има доста али они међусобно не комуницирају, тачније не утичу једни на друге. Свака целина засебно утиче на измене параметра модела током тренинга.

До сада смо дефинисали како се чисте подаци, “токене”, “беч”, тренинг, валидационе и тест скупове, схватили смо да када се подаци очисте, “токенизују” и поделе по тренинг, валидационим и тест скуповима, да се деле по “бечевима” унутар тих тренинг, валидационих скупова, међутим нисмо дефинисали како се ти подаци пакују унутар “бечева”. Језички модели уче на основу секвенца “токена”. Податак типично чини низ “токена” задате величине који служи у предвиђању следећег “токена” у низу. Низ “токена” које користимо за предвиђање називамо карактеристикама (атрибутима, обележјима), док “токен” који желимо предвидети називамо лабелом. Дужину низа карактеристика или лабела можемо мењати, ово је још један од променљивих параметара попут величине “беча” и типично га називамо величина “блока”. Ради бољег разумевања узмимо за пример “беч” величине четири са “блоком” величине осам где су “токени” речи из неког текста. Оваква поставка подразумева да један “беч” за тренирање је дводимензионална матрица величине четири са осам где је свако поље табеле једна реч. Ова матрица је једна од многих матрица карактеристика намењена за тренинг модела. Оно што би оне требале предвидети је исто “беч” лабела који чини матрицу величине четири са осам где свако поље табеле представља следећу реч у секвенци. Матрица лабела је заправо матрица карактеристика померена за један “токен” унапред. На овај начин учимо модел да предвиђа шта треба да следи. Велики језички модели се користе овим техникама за припрему података. Наравно постоје и други приступи. Поменути начин обликовања података је надгледани језички модел где су нам познате и карактеристике и лабеле. Постоје и други приступи попут ненадгледаног модела где су познате само карактеристике, али не и лабеле. Овакав тип модела се може користити у допуњавању текста порука.

### 2.3.2. Дефинисање архитектуре модела

Архитектура модела подразумева распоред, подешавање и избор слојева неуронске мреже. Дубоке неуронске мреже које се користе у великим језичким моделима се састоје од слојева. Слојеви су градивне компоненте дубоке неуронске мреже. Они на улаз модела (које чине бројеви) примењују математичке функције како бисмо као излаз модела добили жељени резултат. У овом случају код великих језичких модела жељени излаз чине вероватноће за сваки “токен” тако да “токен” који треба да следи у низу има највећу вероватноћу. Сам распоред ових слојева, (математичких функција) које ће они применити на улаз модела као и сами параметри који се користе као константе у овим математичким функцијама чине архитектуру модела. Не постоји само једна поставка модела која универзално ради добар посао на сваком скупу података, нити су већ постојане архитектуре једини приступ овом проблему, али постоје пар типова модела који су се показали као добри за задатак језичких модела. Већ поменута трансформер архитектура се показала као веома добар избор за тренинг великих језичких модела и у овом раду ће бити више речи шта сачињава трансформер.

Поред трансформера постоје генерациони модели проширени системом за проналажење информација (РАГ) који повећавају способности великих језичких модела укључивањем страних података из реалног времена омогућавајући корисницима да добију најновије информације. Фалкон језички модели који су уско специјализовани за употребе у доменске или пословне сврхе дају добре резултате у медицини, легалним техничким, финансијским срединама. O претходно поменутим као и изостављеним специфичним архитектурама се овај рад неће бавити, већ ће се фокусирати на општи састав великог језичком модела и трансформер архитектуру.

### 2.3.3. Тренинг

Следећи корак у изради једног великог језичког модела је тренинг. Тренинг модела подразумева покретање рада модела над очишћеним припремљеним подацима. Овај поступак се састоји од иницијализације, пропагације унапред, рачунања губитка (трошка), пропагације уназад и ажурирања параметара модела. Иницијализација модела подразумева насумично или детерминисано подешавање тежина модела. Тежине су присутне у сваком слоју модела, оне представљају коефицијенте којима се улаз модела множи и они су параметри које модел учи како би улаз модела могао да генералише излаз који захтевамо. Цео процес тренинга можемо схватити као процес подешавања тежина, тачније процес промена вредности ових коефицијената које називамо тежинама како би оне за задате улазе давале низ вероватноћа какав очекујемо. У нашем случају ове коефицијенте прилагођавамо на сваком слоју модела тако да када моделу проследимо улазне податке који су низ “токена” представљени у бројевном облику добијемо вероватноће за сваки могући “токен” вокабулара нашег модела где би идеалистично “токен” који треба да следи у низу имао највећу вероватноћу. Ове бројевне репрезентације наших “токена” се множе са тежинама, додатно моделу се може додати и померај како модел не би имао нуле на излазу за неку од вредности тежина. Процес множења улаза модела са тежинама се назива пропагација унапред. Након овог процеса следи рачунање губитка модела. Губитак модела представља разлику између онога што је наш модел предвидео и онога што заправо треба да буде, тачније разлике вероватноћа ова два. У нашем случају би то представљало разлику у претпоставци следећег “токена” у низу нашег модела и “токена” који заправо треба да следи.

За меру разлике ових вероватноћа се често користи унакрсна ентропија:

Унакрсна ентропија = ,

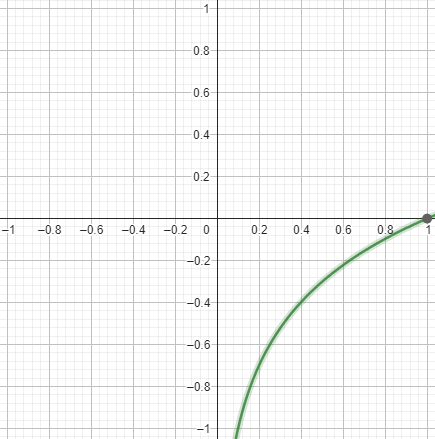
где је:

N – број класа, у нашем случају “токена”,

– права вероватноћа за “токен” *i,*

– претпостављена вероватноћа за “токен” *i.*

Што је већи промашај у нашим претпоставкама то ће укупни губитак бити већи.



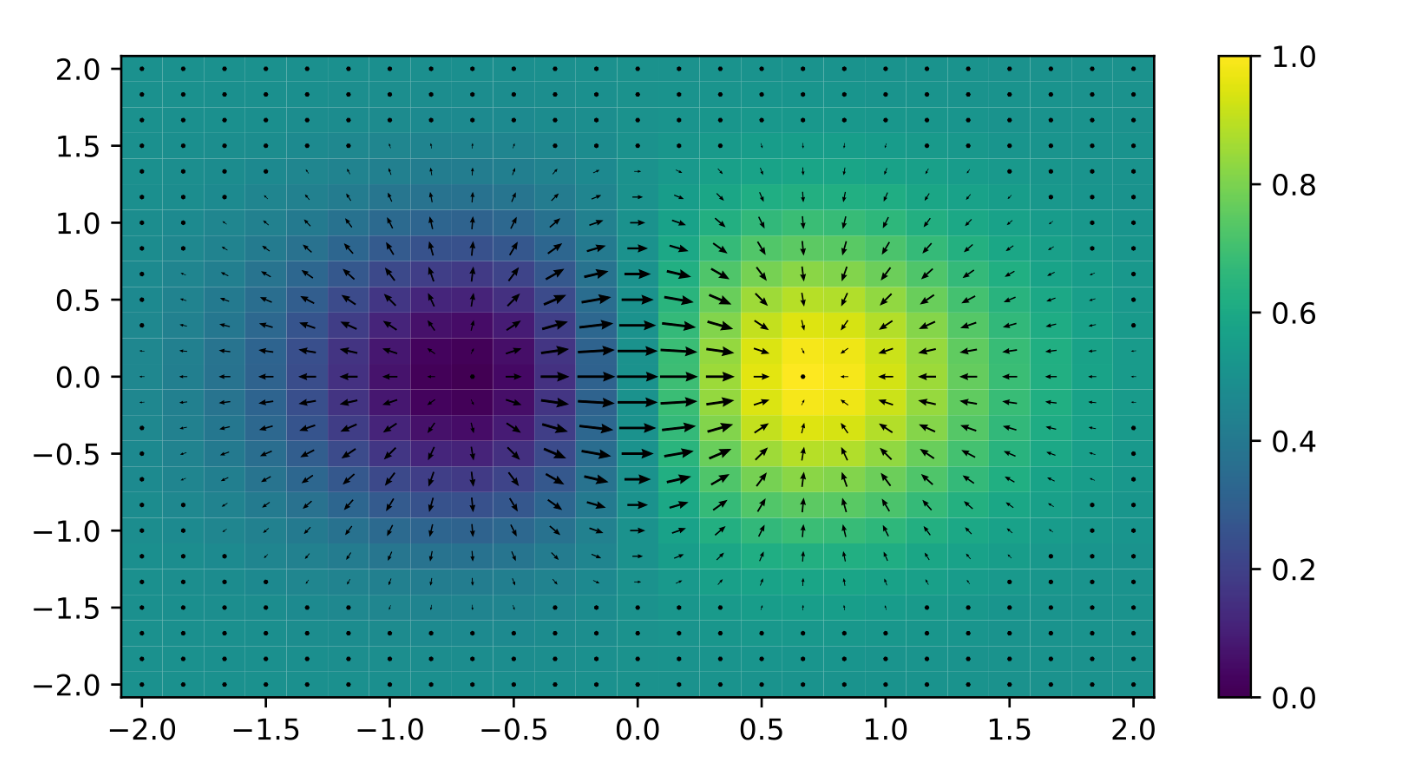
Слика 4. Функција *f*(*x*) = log(*x*)

На Слици 4 можемо видети како се вредност претпоставке нашег модела мења у функцији унакрсне ентропије. Функција “кажњава” модел када је сигуран у претпоставци погрешног “токена”. Након рачунања губитка следи пропагација уназад која подразумева рачунање градијента губитка у односу на сваку тежину. Градијент је вектор скаларних вредности које представљају парцијални извод функције у односу на тежину. Извод нам даје осетљивост излаза функције у односу на њен улаз, тачније количински колико се излаз мења у односу на промену улаза. Он представља нагиб тангетне линије у односу на граф функције у датој тачки. С друге стране градијент се састоји из доста извода и говори нам правац и стопу највећег повећања. Пошто желимо да постигнемо што мањи губитак нашег модела кретаћемо се ка негативној вредности градијента, тачније супротно правцу највишег раста. Градијент функције у тачки се може дефинисати као:

где је:

– *n* димензионални вектор,

– парцијални извод функције у тачки у односу на тежину .

Слика 5. 2Д приказ градијента за функцију (извор: [4])

На Слици 5 можемо видети како градијент изгледа у 2Д простору за насумичну функцију. Стрелице су усмерене ка већим вредностима, наш модел иде у супротном правцу, ка мањим како би смањио губитак. На крају вредности тежина се ажурирају у супротном правцу од градијента. Kолики корак ће направити зависи од стопе учења коју задајемо моделу [18]. За ажурирање тежина се користе оптимизациони алгоритми. Овај процес се понавља задани број пута и назива се број епоха где сам круг описаних операција потребних за једно ажурирање тежина се назива епоха.

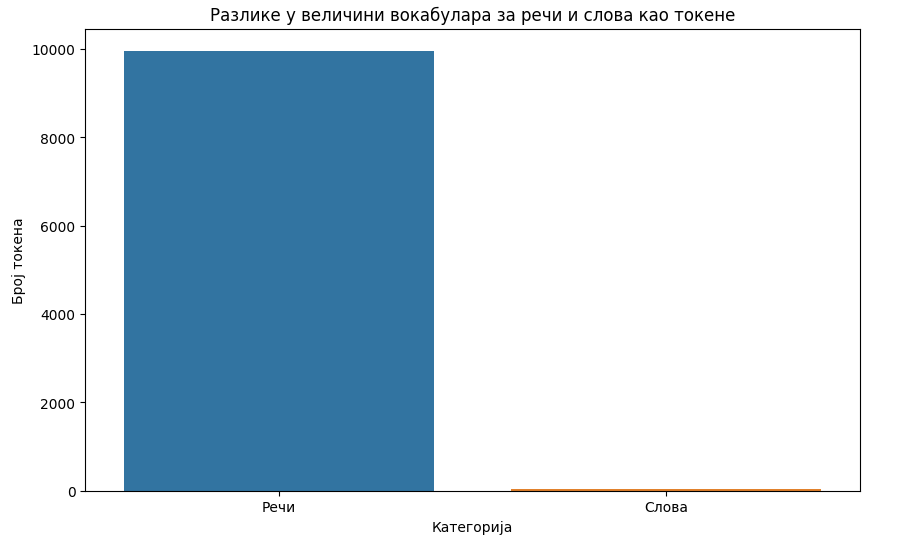
### 2.3.4. Евалуација

Евалуација модела се врши посматрањем губитка модела, међутим губитак некад може да завара процену модела тако да је некад важније да ли модел даје жељене излазе иако је губитак већи него очекиван.

# 3. Имплементација

## 3.1. Чишћење и обрада података

За тренинг великог језичког модела одабрана је књига Медитације Марка Аурелија. Цела књига се учитава, а потом прави подела са два избора “токена”. Први избор су слова као “токени”, а друга речи као “токени”. Што је број карактера који “токен” садржи већи, то граматика садржи више индивидуалних “токена”, а секвенца токена којим енкодирамо ове “токене” илити величина “блока” постаје мања. Наиме избор “токена” диктира комплексност граматике па самим тиме и временско трајање тренирања модела јер модел мора да тренира већи број тежина. Још једна напомена је да са дужим “токенима” добијамо већу прецизност на нивоу “токена”, тачније код избора слова за “токене” можемо добити несмислене конструкције речи. На Слици 6 можемо видети разлике у величини вокабулара са одабиром речи и слова као “токена”.



Слика 6. Разлике у величини вокабулара за речи и слова узете за токене књиге Медитације Марка Аурелија

Вокабулар слова је очишћен тако да садржи сва слова алфабета уједно са специјалним симболима попут размака, тачке, узвичника и упитника. Вокабулар са речима је очишћен од речи које садрже бројеве. Ово могу бити појаве фуснота у књизи, а поред тога су свим речима смањена слова како би се избегли дупликати. Након овога за вокабуларе су направљени једностаџвни енкодери и декодери у виду ламбда функцијe која додељујe сваком “токену” редни број. Цео текст је енкодиран у једнодимензионални тензор (аналогно матрици), а затим подељен тако да осамдесет посто текста припада тренинг скупу и преосталих двадесет служе за валидацију.

## 3.2. Биграм модел

Први тип модела је једноставни биграм модел. Биграм модел подразумева тренирање 2Д ембединг матрице. Ембединг речи чини њено представљање у математичком векторском простору. У овом простору вектори речи сличног значења се налазе близу једни других. На овај начин се бележе подаци семантичког значења речи и моделу омогућава учење релација међу њима. Матрица је величине вокабулара са величином вокабулара. Свако поље матрице представља лоџит. Лоџит је вредност чијом обрадом добијамо вероватноће просторне појаве неког “токена” после другог. Лоџити су необрађене бројчане вредности које нормализацијом постају вероватноће. Модел се састоји из ручно имплементиране функције пропагације унапред која за дати “токен” рачуна разлике лоџита “токена” које модел претпоставља да треба да следе са лоџитима “токена” које заправо следе. Разлике ових лоџита чине нумерички промашај модела и уједно трошак који се рачуна преко већ поменуте унакрсне ентропије. У самој имплементацији није примењена софтмакс функција пре унакрсне ентропије јер њена имплементација у “торч” модулу који је коришћен већ то аутоматски ради.

### 3.2.1. Софтмакс активациона функција

Поред овога модел садржи функцију генерисања “токена” која за задати улаз узима лоџите провлачењем улазних података кроз функцију пропагације унапред. Добија вероватноће последњег токена у низу провлачењем ових лоџита кроз софтмакс активациону функцију, потом извлачи примерак са највећом вероватноћом, тачније извлачи “токен” са највећом вероватноћом појаве у низу и додаје га на тренутни улаз, илити на претходне токене. Овим поступком се генерише текст са биграм моделом. Софтмакс функција се може дефинисати као:

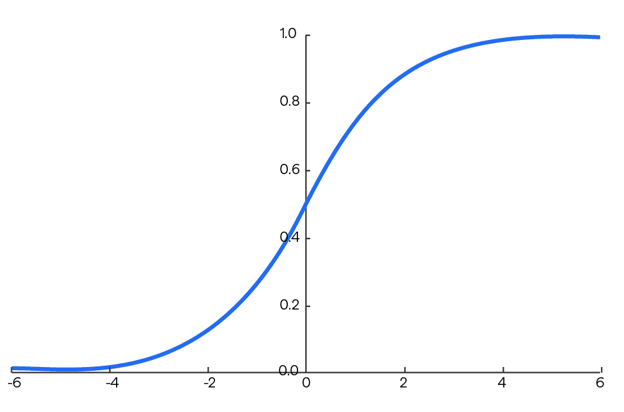
где је:

– вектор дужине *k,*

– сума вредности свих *k* експонената вредности вектора *z,*

– експонент вредности на позицији *i* вектора *z.*

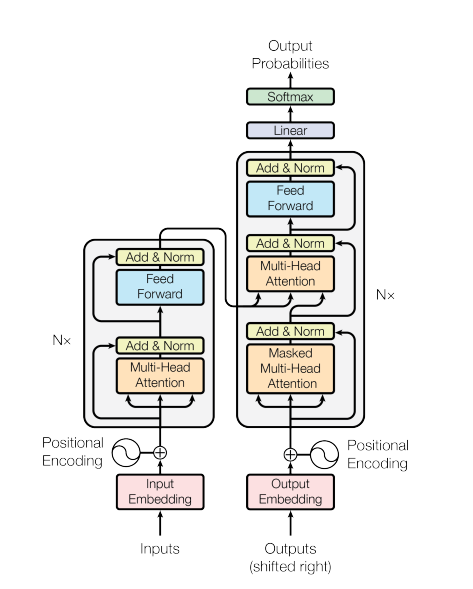
Софтмакс функција трансформише излазе нашег модела у расподелу вероватноћа, ово значи да сви излази нашег модела заузимају вредности између 0 и 1 и у збиру дају 1. Друге предности софтмакс функције јесу да се разлике између класа боље одражавају. Модел је одлучнији у својим изборима и вредности су лако разумљиве, моделу је лакше радити са мањим вредностима на одређеној скали. Промену излаза софтмакс функције у односу на улаз можемо видети на Слици 7. Можемо видети да се излази благо додељују.



Слика 7. Граф софтмакс функције (извор: [5])

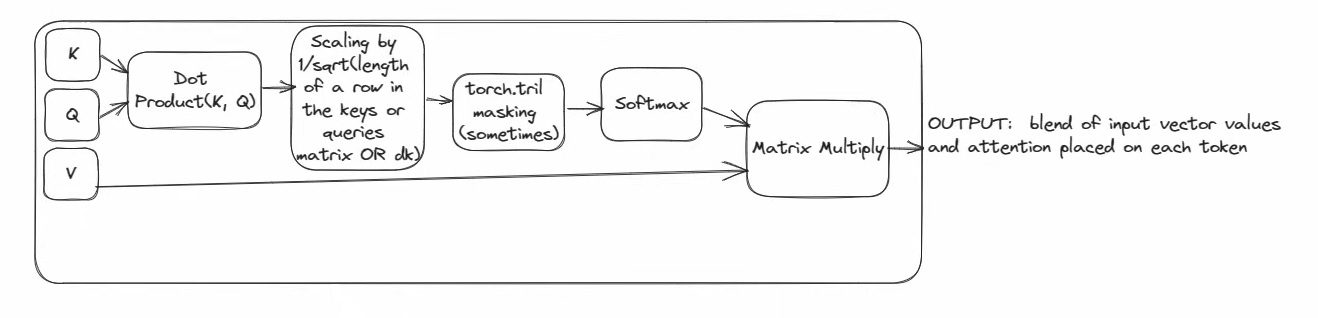
## 3.3. ГПТ модел

ГПТ илити генеративни претренирани трансформер веће комплексности архитектуре уврштава и механизам пажње. За разлику од једноставног биграм модела, ГПТ покушава да обрати већу пажњу претходном контекту, где је биграм модел само способан да предвиди који “токен” следи у секвенци у односу на први претходни “токен”. ГПТ ово постиже конструкцијом коју називамо глава. На Слици 8 можемо видети архитектуру трансформера из рада *“Attention is all you need”* [6]која је заслужна за настанак данашњих великих језичких модела па и ГПТ – а.



Слика 8. Архитектура трансформер модела (извор: [6])

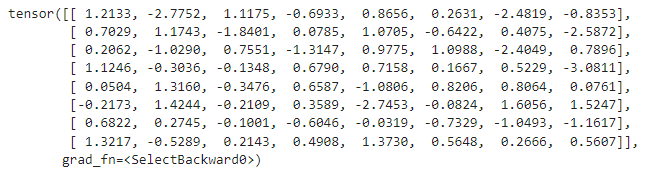
### 3.3.1. Трансформер глава



Слика 9. Глава ГПТ модела где су: K – кључ, Q – упит, V – вредност

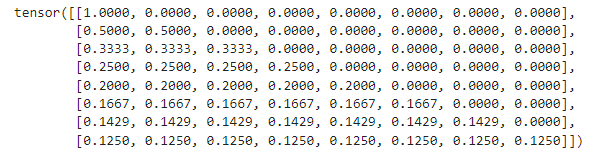
(извор: [7])

Глава је једна од најбитнијих градивних целина ГПТ модела. Идеја је да пратимо вредности за све претходне токене “блока”. Подсетимо се “блок” је низ токена које у једном тренутку дајемо моделу за предвиђање следећег “токена” у низу. Да бисмо пратили ове вредности могли бисмо се послужити усредњавањем вредности “токена”, међутим ово не би забележило однос и афинитете неких “токена” у односу на друге. Глава ГПТ модела решава овај проблем на следећи начин, она емитује две матрице, упит и кључ. Идеја ове две матрице је да упит садржи информације онога што ја (“токен”) тражим, а кључ представља шта ја садржим. Ред ове матрице представља један “токен”, док колона представља карактеристику токена која се у наставку спомиње. Ове две матрице настају провлачењем ембединг матрице са улаза кроз линеарне слојеве модела који мапирају вредности (редимензионишу ембединг матрицу) у број карактеристика, илити величину главе. Афинитете токена добијамо производом његове матрице упита са матрицама кључева свих осталих “токена” уколико ситуацију посматрамо као један “токен”. У глави постоји параметар који одређује величину главе. Овај параметар је само цели број о којем можемо интуитивно размишљати као број карактеристика које “токен” може да има. За неке карактеристике ће имати већи број него за друге и уколико неки други токен тражи ову карактеристику (има велики број за ту карактеристику у свом пољу упит матрице) тада ће ова два токена имати јачу везу, тачније већу шансу да се појаве један за другим током процеса генерације новог токена јер ће и сам производ бити већи. На Слици 9 можемо видети визуелни приказ главе трансформер модела.



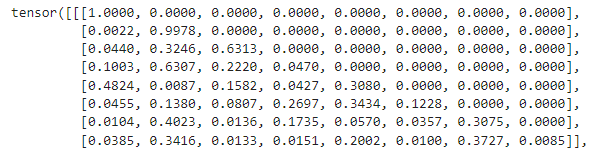
Слика 10. Насумична матрица пажње за “блок” величине осам

Дакле да рeзимирамо, гледајући једну ставку “беча” глава ће генерисати матрицу “блок” величине са “блок” величином множењем кључева и упита јер желимо да знамо афинитете за сваки токен у секвенци у односу на друге токене. Цела матрица се дели кореном величине друге димензије улазне ембединг матрице ради нормализације. На Слици 10 можемо видети пример“беча” где су редови “блокови”. Додатак овоме је да на почетку генерисања модел нема контекст, такође модел приликом тренинга не би требало да има појам о будућности. Простије речено “токен” на позицији три не би требало да зна шта се налази на позицији четири, већ само шта се налази до позиције три. Овај проблем се решава множењем вредности матрице пажње са матрицом чије су вредности испод главне дијагонале једнаке бројевима који у суми реда дају један, а изнад те дијагонале се налазе нуле. Да бисмо добили збир од 1 у редовима доњег левог угла можемо се послужити већ поменутом софтмакс функцијом над поменутом троугаоном матрицом. Пример резултата овог поступка можемо видети на Слици 11.



Слика 11. Матрица за нормализацију вредности матрице пажње и њеног ускраћивања гледања будућности

Када прву матрицу пажње помножимо оваквом матрицом добијамо да наши токени не гледају у будућност приликом тренинга и други бенефит је да су вредности нормализоване и тиме интуитивније и лакше за рад моделу. На Слици 12 можемо видети резултат овог поступка.

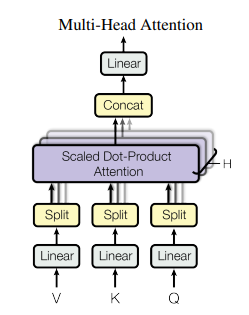


Слика 12. Комплетирана матрица пажње

Комплетирану матрицу пажње можемо помножити ембединг матрицом коју добијамо као улаз у главу, међутим још један корак је потребан. Ембединг матрица са улаза се провлачи кроз линеарни слој неуронске мреже која као и код упита и кључева главе мапира вредности на скали карактеристика “токена” и потом се множи комплетираном матрицом пажње. Ова матрица се уједно назива и матрицом вредности. Крајње поменуто множење овом матрицом се може видети на Слици 9.

### 3.3.2. Пажња више трансформер глава

Позната Вам је изрека две главе су паметније од једне, то важи и код модела вештачке интелигенције. Наиме више глава се пакује у већу целину са још пар додатака. Ову структуру можете видети на Слици 13 [6].

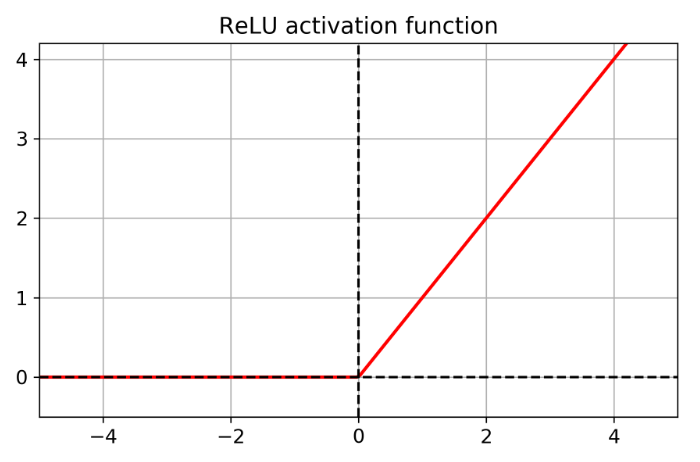


Слика 13. Пажња више трансформер глава (извор: [6])

Овим се постиже да више глава паралелно раде. Улазне податке делимо по последњој димензији, илити димензији канала и потом излазе ових глава надовезујемо. Ово помаже у проналажењу другачијих и разноврснијих података и зависности “токена”. После надовезивања излази се провлаче кроз линеарни слој и дропаут слој о којем ће бити речи у наставку.

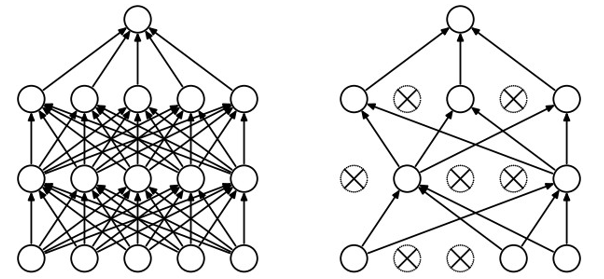
### 3.3.3. Слој пропагације унапред

До сада смо видели како “токени” међусобно “комуницирају”. У главама смо покушали да забележимо податке међусобног афинитета “токена”. Слој пропагације унапред је једноставан вишеслојни перцептрон који индивидуално обрађује сваки податак. Идеја је да се “токенима” омогући појединачна обрада података прикупљених у преходним корацима без утицаја спољашњих фактора (других “токена”). Можемо о овоме интуитивно размишљати као да токени упијају и обрађују придобијено знање. Слој пропагације унапред се редом састоји од линеарног слоја, релу активационог слоја, линеарног слоја и дропаут слоја. Релу, илити исправљачки линеарни слој негативне вредности претвара у нуле, а позитивне оставља какве су биле. Пример ове функције се види на Слици 14 [8].



Слика 14. Релу активациона функција (извор. [8])

Дропаут слој је користан у неуронским мрежама како би се спречило напамет учење тренинг података и научила вештина доброг генерализовања модела. Дропаут слој насумично бира претходно задат процентуални део неурона и анулира њихове вредности, па се у следећи слој прослеђују нуле, тачније ови подаци не учествују у пропагацији унапред [9]. На Слици 15 са леве стране можемо видети густо повезану неуронску мрежу пре примене дропаут слоја и са десне стране учинак дропаут слоја, тачније стрелице које фале су вредности које су анулиране.

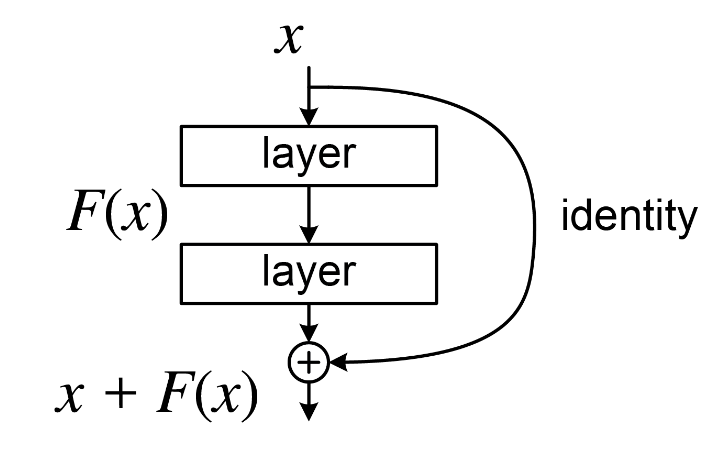


Слика 15. Последице дропаут слоја на неуронску мрежу (извор: [9])

### 3.3.4. Трансформер блок

До сада смо виделу трансформер главу, пажњу са више трансформер глава и слој пропагације унапред, али како се те компоненте склапају у једну целину. За решавање овог проблема се користи трансформер блок. Ова структура се редом састоји од слоја са више трансформер глави, слоја линеарне нормализације, слоја пропагације унапред и поново слоја линеарне нормализације. У ГПТ архитектури овај трансформер блок може бити поновљен већи број пута, за наше потребе тај број је шест. Ако пажљиво погледамо слику 8 можемо приметити блок са десне стране означен симболом *Nx,* додуше наша имплементација ГПТ модела нема енкодер делове са леве стране слике, па самим тиме ни средишњи слој у трансформер блоку *Nx,* тачније немаскиран слој више трансформер глави.

Слојеви линеарне нормализације који се налазе између слоја више трансформер глави пажње и слоја пропагације унапред примају излаз претходног слоја али и улаз претходног слоја, па потом ова два улаза сабирају. Ове везе називамо резидуалним. Приметимо да слојеви нормализације на Слици 8 (назначени жутом бојом) имају стрелицу која води из излаза претходног слоја као и стрелицу са десне стране улаза тог слоја. Ова ставка је битан део трансформер мреже. Као што можемо видети модел је много дубљи и компликованији од биграм модела. Код овако дубоких мрежа може доћи до проблема нестајућих градијената приликом пропагације уназад у тренингу модела. Проблем настаје тиме што градијенти са краја модела за велик број слојева постају изузетно мали не доприносећи пуно у ажурирању тежина у слојевима са мањих дубина [11]. Овај проблем управо покушавају да реше резидуалне везе. Резидуалне везе нису од великог значаја приликом пропагације унапред, али приликом пропагације уназад имају кључну улогу. У пропагацији уназад одржавамо информације градијената дубљих слојева тиме што они споредним резидуалним везама доспевају до слојева мањих дубина [10]. На Слици 16 можемо видети резидуалну везу која прескаче два слоја главне путање и сабира се са њом после та два слоја.



Слика 16. Илустрација резидуалне везе (извор: [10])

### 3.3.5. Функција пропагације унапред

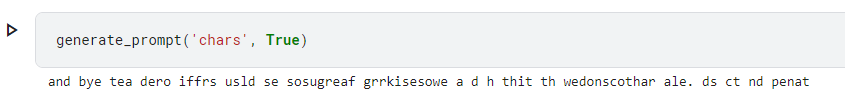
Сада имамо све потребне градивне елементе за фунцију пропагације унапред нашег ГПТ модела. Подсетимо се до сада смо видели трансформер главу, како више трансформер глава паралелно раде на подељеним подацима улаза, слој пропагације унапред и како се све те компоненте склапају у трансформер блок. Подсетимо се и биграм модела. Он је за своје тежине чувао ембединг матрицу тежина “токена”. Код ГПТ модела имамо и додатак матрицу исте величине ембединг матрице тежине “токена” намењену за позиционе податке “токена”. Првобитна матрица тежина “токена” намењена је да енкодира идентитете “токена”. Новом матрицом енкодирамо податке позиције “токена”. Сама функција пропагације унапред трансформера је једноставна након разумевања свих досадашњих градивних елемената. Матрице идентитета и позиција “токена” се сабирају, потом се провлаче кроз слојеве блока трансформера. У нашем случају кроз шест трансформер блокова и потом кроз линеарни слој који их мапира назад на величину вокабулара. Сада поседујемо лоџите за “токене” нашег вокабулара. Ови лоџити се пореде са вредностима које заправо треба да следе илити лабелама употребом већ поменуте унакрсне ентропије. Излаз функције пропагације унапред нашег модела су лоџити и губитак. Функција за генерисање токена је иста као код биграм модела. На улаз се прима матрица секвенце “токена”, одсеку се “токени” на позицијама у прошлости већим од величине блока (овде се говори о блоку “токена”, тачније број “токена” који служи за предвиђање следећег “токена” у низу), лоџити и губитак се добијају из горе поменуте функције пропагације унапред. Вероватноће се добијају провлачењем лоџита кроз поменуту софтмакс функцију ради нормализације и затим се бира највероватнији следећи токен у секвенци.

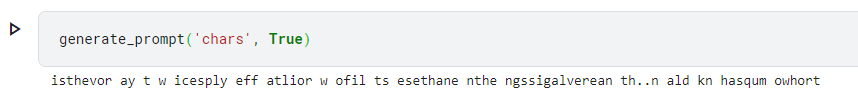
# 4. Резултати и дискусија

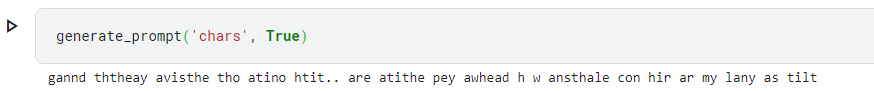
За потребе овог рада имплементиране су две архитектуре модела, једноставни биграм модел и комплексни трансформер модел. Поред овога направљене су и две варијанте где су се користила слова и речи за “токене”. Вокабулар са словима се састојао од 30 “токена”, док се модел са речима састојао од 9959 “токена”.

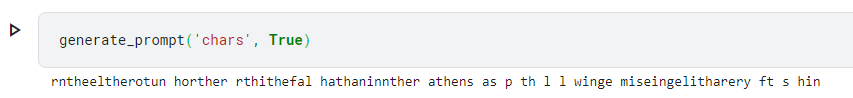
## 4.1. Биграм модел слова

Варијанта са словима као “токенима” је са мало већим бројем итерација давала резултате који су покушај речи. Модел је почео да прави размаке на местима која изгледају тачна, и помислили бисте да генерише смислен текст уколико бисте мало зажмурили и одмакли се од монитора, међутим модел није могао да забележи комплексности језика чак ни са вокабуларом од тридесет карактера. Модел је највише могао да коректно наређа 3 до 4 “токена” заредом што се може видети у првом примеру са слике 17 речи “and” и “bye”.



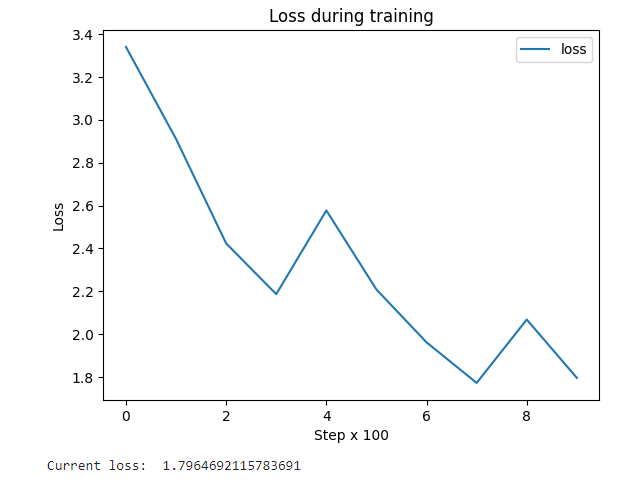






Слика 17. Примери генерисања “токена” биграм модела са словима као “токенима”

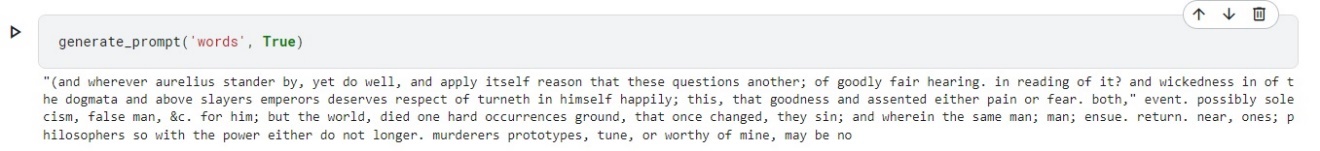
Ради побољшања модела имплементирана је функција која је више пута тренирала модел са по 1000 епоха мењајући параметре стопе учења и величине “блока” (говоримо о броју “токена” који модел добије у датом тренутку) и поред овога цртала графике стопе учења у односу на епохе. Наравно промена величине “блока” има смисла само код трансформер модела јер биграм модел предвиђа наредни “токен” у секвенци само у односу на претходни. Модел је релативно стабилно смањивао губитак . Контраинтуитивно већа стопа учења је била повољнија за овај модел. Експериментима је утврђено да је измећу једне и три децимале испод нуле оптимална величина стопе учења. Meњање параметара “беч” величине није имало великог утицаја на биграм модел. Боље резултате генерисања токена имао је модел са већом стопом учења од 0,01. Ако погледамо слику 17 можемо сагледати примере генерисања токена овог модела. Смањивање стопе учења је давало лоше резултате, претпоставка је да је модел упадао у локални минимум. Губитак приликом тренинга током епоха се може сагледати на графу слике 18. На *x* оси се налази редни број стотине епохе, а на *y* оси губитак.

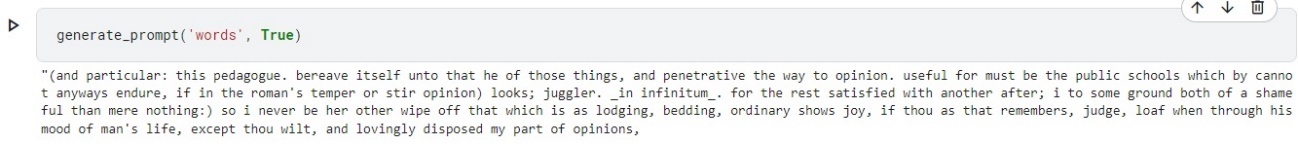


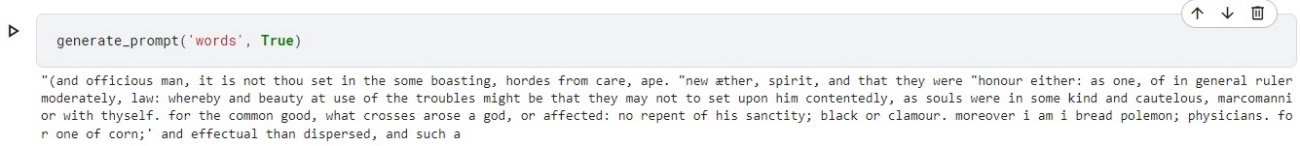
Слика 18. График губитка у односу на епохе биграм модела са словима за “токене”

## 4.2. Биграм модел речи

Биграм модел речи имао је сличну појаву бољег тренинга са већом стопом учења. Занимљиво је приметити разлику у односу на биграм модела са словима, модел речи је успевао да смислено повеже две до пет речи у секвенци. На Слици 19 можемо видети неке резултате генерисања токена овог модела. Као контекст је дата реч “and” на почетку па се свуда и појављује.

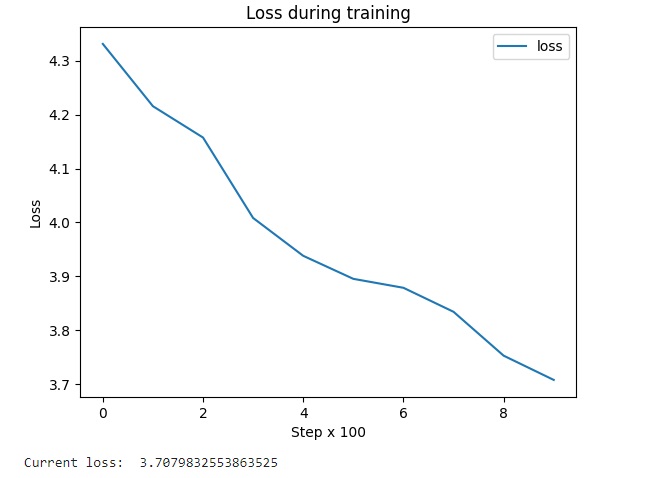






Слика 19. Генерисане реченице биграм модела са речима као “токенима”

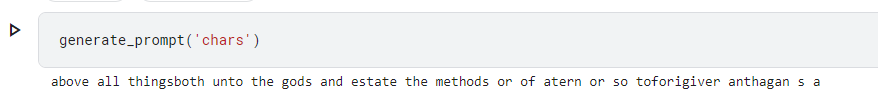
Главне разлике ова два приступа су да модел речи има већи вокабулар, па самим тиме тренинг оваквог модела за исти број епоха траје дуже него биграм модел слова. Друга значајна напомена је да се биграм модел речи бави самим организовањем речи и нема исти проблем бесмисла на нивоу речи као што је то случај код биграм модела слова. Питање које се поставља је како треба поступити када се код оваквог модела на улаз доведе вредност коју модел нема у корпусу вокабулара. Тренутна имплементација модела само одбацује такав “токен” као да се не налази у секвенци. Још једна разлика је да због величине самог вокабулара, тежине овог модела заузимају много већи меморијски простор. Биграм модел због своје једноставне архитектуре није погодан за прављење озбиљног језичког модела који би могао да генерише дуге секвенце, али за потребе програма за аутоматско допуњавање порука и сличних примена се чини поприлично добрим избором. Једноставна архитектура је оно што га издваја од трансформер модела и мања меморијска величина. Тренинг је доста бржи и захтева мање рачунарских ресурса. Обе варијанте модела су трениране са 1000 епоха, понекад са поновним покретањем тренинга. Губитак приликом тренинга током епоха се може сагледати на графу слике 20. На *x* оси се налази редни број десетине епохе, а на *y* оси губитак.

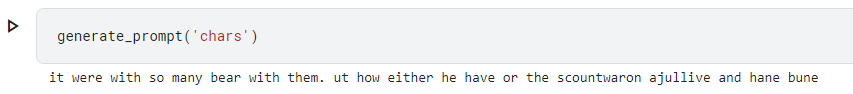


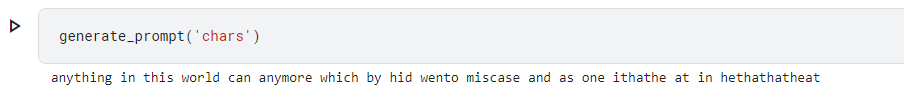
Слика 20. График губитка у односу на епохе биграм модела са речима за “токене”

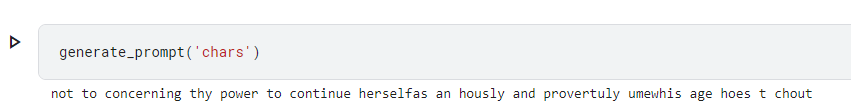
## 4.3. Трансформер модел слова

Трансформер модел је компликованији од биграм модела, па су и сама подешавања другачија. Додатна подешавања величине главе којих је коришћено 6, број трансформер блокова којих је исто коришћено 6 и број ембединг димензија којих је коришћено 384. Величина блока је 256 и величина “беча” је 64. Тренинг је рађен са 1000 епоха и покретан више пута, а стопа учења је 3×10-4. Процес тренирања је доста дужи него код биграм модела. Један циклус од 1000 епоха је трајао око 15 минута. Може се приметити да је трансформер модел слова значајно бољи у односу на биграм модел што се тиче прецизности. Речи имају смисла, док реченице не у потпуности. На Слици 21 можемо видети резултате генерисања трансформер модела слова. Модел даје добру понуду наставка текста у дужини од 3 до 5 речи у односу на последња уписана слова. Тренутно првих 80 – так токена изгледају прихватљиво. Токени су генерисани са датим празним контекстом. На Слици 22 се може видети однос губитка према стотинама епоха у једном покренутом тренингу.

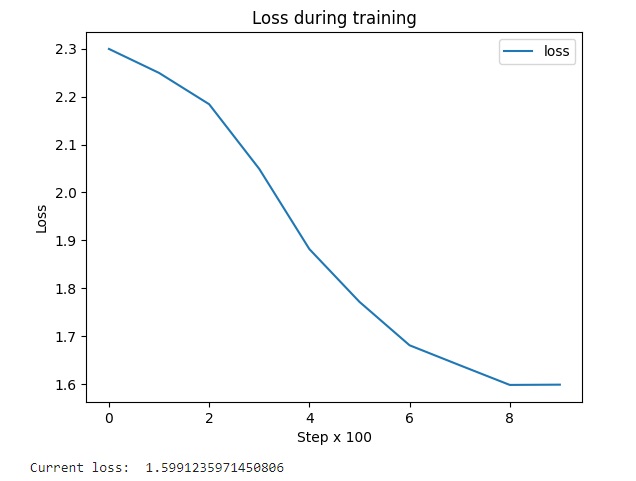








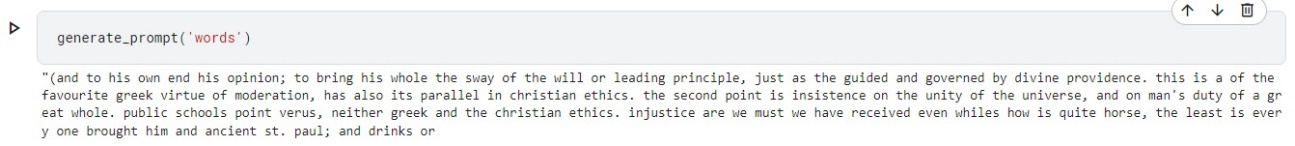
Слика 21. Генерисане речи трансформер модела са словима као “токенима”

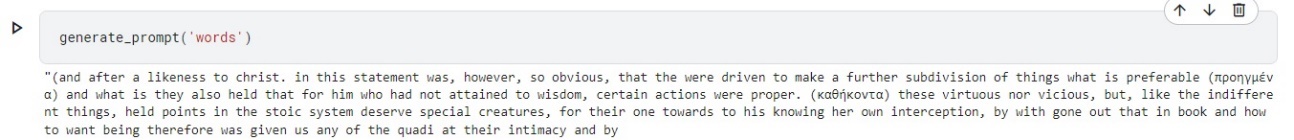


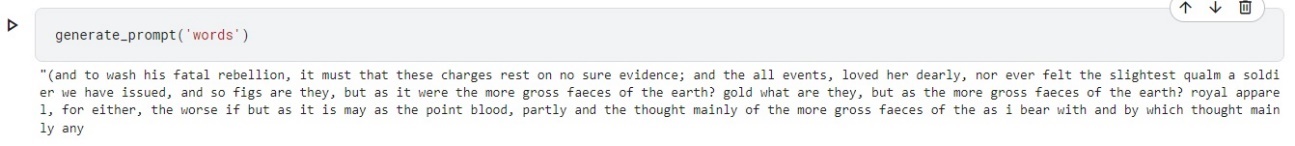
Слика 22. График губитка у односу на епохе трансформер модела са словима за “токене”

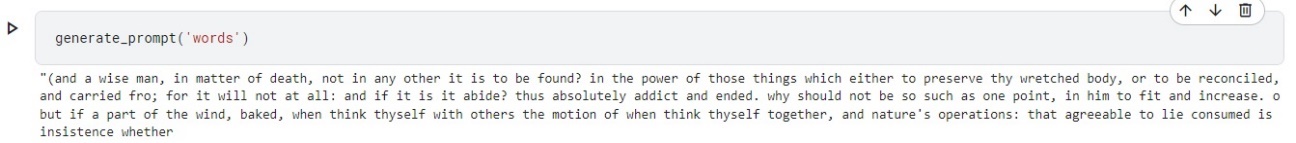
## 4.4. Трансформер модел речи

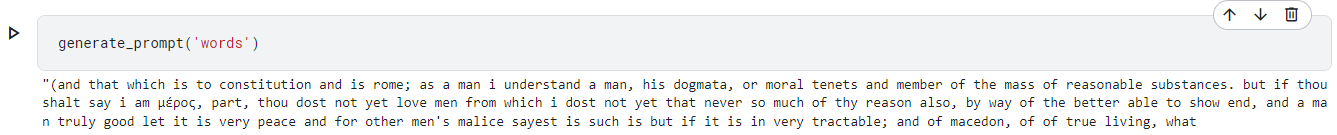
Резултати транформер модела речи су били најбољи. Модел је почео да бележи комплексност текста и да даје смислене одговоре на промпт са само контекстом речи “and”. Претпоставка је да је модел дао смислене резултате фокусирајући се на распоред самих речи уместо слова. Тренинг је рађен са истим подешавањима као и код трансформер модела слова. На Слици 23 се могу видети запањујући резултати овог модела. Модел није перфектан, али се могу видети занимљиве смислене целине. У неким одређеним деловима реченице недостаје по нека реч. На Слици 24 се може видети стабилно смањење губитка кроз стотине епоха у једном покренутом тренингу.



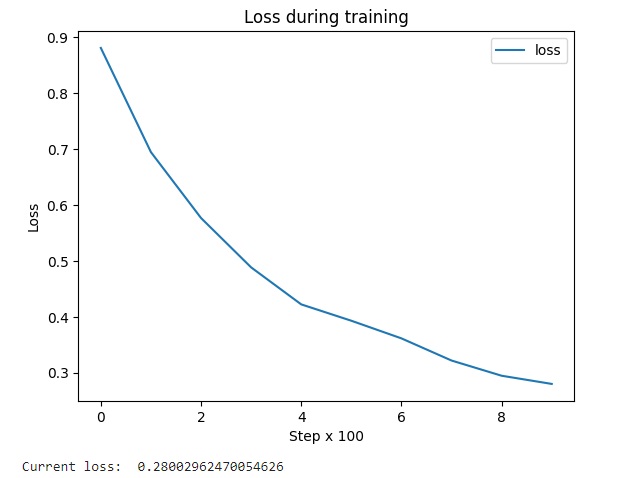








Слика 23. Генерисане реченице трансформер модела са речима као “токенима”



Слика 24. График губитка у односу на епохе трансформер модела са речима за “токене”

# 5. Закључак

У овом раду је приказан поступак изградње два типа великих језичких модела са две варијанте за сваки тип понаособ. Изложени су детаљи огранизације неуронске мреже великих језичких модела и њени мотиви. Направљено је поређење ових типова модела. Рад покушава да приближи сложеност архитектуре језичких модела актуелних у моменту писања овог рада. Биграм модели су показали своје квалитете једноставности имплементације и кратког времена тренирања. Погодни су за једноставније програме допуне и могли би се користити у варијанти са речима као “токенима”. С друге стране трансформери, иако компликовани, долазе до изражаја у сналажењу са комплексним релацијама података реалног света. Добро бележе зависности и могу да се изборе са сложеним проблемима попут генерисања текста природног језика.

Добијени добри резултати трансформер модела речи отварају бројне могућности за даље истраживање. Идеја за побољшање модела би била поделити улазни текст на низ питања или саопштења са одговорима из датог текста. На овакав начин бисмо научили модел да даје стоичке савете на људске проблеме. Идеја је дати моделу примере на основу којих би се сналазио у таквом контексту. Сличне идеје су поменуте у раду [12]. Поред ових измена могуће промене параметара модела као и додавање додатних извора података би дала могућа значајна побољшања имајући у виду да је модел трениран само на тексту једне књиге.

Од релативно скоријег времена 2012. године када је једна мрежа применила идеје ранијих научних истраживања и тиме успела да обради комплексне слике реалног света звана АлексНет [19], чини се да су модели рађени да подрже све већи број параметара. Објављивањем рада “*Attention Is All You Need”* [6] 2017. годиненастали су многи велики језички модели, али се чини као да се више ради на подржавању већег броја тежина, архитектура се није значајно променила. Можда баш у овоме лежи сакривени пут до једноставнијег, прецизнијег и мање скупог модела. Недостаје приступ који би комбиновао једноставност биграм или сличних модела са комплексним способностима трансформер модела дубоким разматрањем иновативније архитектуре уместо претрпавања постојећих са огромним бројем параметара.

# 6. Литература

1. Paul Rogers. (2023). 56 ChatGPT Statistics For 2024 (Users, Facts & Trends). https://www.answeriq.com/chatgpt-statistics/
2. Tuana Çelik. (2023). The Definitive Guide to BERT Models. https://www.deepset.ai/blog/the-definitive-guide-to-bertmodels
3. Rob Williams. (2017). NVIDIA GeForce GTX 1080 Ti Review: A Look At 4K & Ultrawide Gaming. https://techgage.com/article/nvidia-geforce-gtx-1080-ti-review/
4. URL: https://commons.wikimedia.org/wiki/File:Vector\_Field\_of\_a\_Function's\_Gradient\_imposed\_over\_a\_Color\_Plot\_of\_that\_Function.svg
5. URL: https://botpenguin.com/glossary/softmax-function
6. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. (2017). Attention Is All You Need. arxiv. https://arxiv.org/abs/1706.03762
7. Elliot Arledge. URL: https://github.com/Infatoshi/fcc-intro-to-llms & https://www.youtube.com/watch?v=UU1WVnMk4E8&t=10397s
8. Sovit Ranjan Rath. (2019). Activation Functions in Neural Networks. https://debuggercafe.com/activation-functions-in-neural-networks/
9. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf
10. Ngieng Kianyew. Diving into Residual Connections. (2023). https://medium.com/@ngiengkianyew/residual-connection-intuition-c8cf011170d
11. Sabyasachi Sahoo. Residual blocks — Building blocks of ResNet. (2018). https://medium.com/towards-data-science/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
12. Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei. Language Models are Few-Shot Learners. (2020). arxiv. https://arxiv.org/pdf/2005.14165
13. URL: https://www.deeplearning.ai/resources/natural-language-processing/
14. URL: https://aws.amazon.com/what-is/large-language-model/
15. URL: https://aws.amazon.com/what-is/nlp/
16. Keith D. Foote. A Brief History of Large Language Models. (2023). https://www.dataversity.net/a-brief-history-of-large-language-models/
17. Michael McDonough. large language model. (2024). https://www.britannica.com/topic/large-language-model
18. Simeon Kostadinov. Understanding Backpropagation Algorithm. (2019). https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd
19. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks . (2012). https://proceedings.neurips.cc/paper\_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf