УНИВЕРЗИТЕТ У БЕОГРАДУ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Евалуација финансијског агента користећи велики језички модел као судију

Дипломски рад

Ментор
проф др. Милош Цветановић

Студент
Алекса Рачић, 2023/3001

Београд, септембар 2025

Садржај

Увод

1.1 Значај финансијских текстуалних података

Савремени финансијски екосистем генерише огромну количину текстуалних података — од вести у реалном времену и аналитичких извештаја до регулаторних пријава и објава на друштвеним мрежама. Ови неструктурирани текстови носе кључне увиде о тржишним условима и основама пословања компанија, који често утичу на перцепцију и одлуке инвеститора.

Процењује се да најмање 80% свих данашњих података чине неструктурирани подаци, што обухвата и наведене финансијске текстуалне токове [1]. Суочавање са овим таласом информација постало је озбиљан изазов: огроман број дневних финансијских вести и извештаја једноставно је немогуће да било који човек у целости прочита и обради.

Важни детаљи лако могу промаћи када су аналитичари затрпани документима од стотину страница или непрекидним током вести. Због тога је неопходно развијати алате и технике који помажу разумевању и извлачењу увида из великих количина финансијског текста.

Ефикасна анализа ових текстуалних извора критична је не само за инвеститоре и аналитичаре који желе да доносе информисане одлуке, већ и за регулаторе и истраживаче који се ослањају на квалитативне информације које сирови квантитативни подаци не могу да обухвате.

1.2 НЛП и велики језички модели мењају анализу финансија

Напредак у машинском учењу, а посебно у обради природног језика (НЛП), драматично је променио начин на који финансијска индустрија обрађује текстуалне податке.

НЛП омогућава рачунарима да тумаче и уносе структуру у неструктуриран текст, претварајући квалитативне информације у квантитативне сигнале или сажетке који су знатно лакши за анализу.

На пример, задаци који би за човека били изузетно временски захтевни — преглед хиљада новинских чланака ради процене сентимента, читање транскрипата позива поводом зарада ради идентификације кључних тема, или поређење језика у више годишњих извештаја — сада се могу обавити у релативно кратком времену.

НЛП алати могу брзо да обраде масивне количине текста како би уочили трендове, измерили сентимент и истакли потенцијалне ризике, суштински претварајући људски језик у примењива сазнања [2].

Кроз рударење и анализу великих текстуалних корпуса, савремени НЛП системи помажу у подршци пословним процесима, откривању макроекономских сигнала и побољшању доношења одлука у финансијским институцијама [3].

Велики скок у способностима НЛП-а донели су велики језички модели (LLM). То су дубоки модели тренирани на огромним корпусима текста, који омогућавају висок ниво разумевања језика и генерисања текста.

Модели попут GPT-4 показали су изузетну дубину разумевања, у стању да интерпретирају нијансе финансијског језика и контекста након обуке на великим скуповима текстова [2].

За разлику од ранијих НЛП система који су често захтевали специфично дотренирање за сваки задатак, LLM-ови могу да решавају широк распон задатака уз минимално или без додатног тренирања, захваљујући општем језичком и светском знању које су усвојили.

У финансијском домену ови модели могу да произведу организоване, експертске анализе сложених докумената, идући даље од површинске читљивости ка разумевању суптилних детаља и доменске терминологије [3].

Један упит LLM-у може да да сажетак годишњег извештаја од 100 страница или одговори на детаљна питања о његовом садржају, практично симулирајући рад искусног финансијског аналитичара.

1.3 Улога 10-K извештаја као кључних финансијских докумената

Међу бројним текстуалним изворима у финансијама, годишњи 10-K извештаји које подносе америчке јавне компаније издвајају се као посебно важни и информативни. Комисија за хартије од вредности САД (SEC) налаже да компаније поднесу 10-K, свеобухватне годишње извештаје који дају детаљан приказ финансијских резултата и пословних активности фирме [3]. Типичан 10-K садржи ревидиране финансијске извештаје, уз опширне наративне сегменте у којима се разматрају пословни модел и стратегија, тржишни услови, фактори ризика. У суштини, он пружа холистичну и структуирану слику рада и планова компаније. Кључно је да ова документа укључују и квалитативне

информације — попут објашњења стратегије и процена ризика — које квантитативне метрике саме по себи не могу да ухвате.

Ипак, 10-К извештаји су озлоглашено обимни и сложени, често имајући стотине страница густог текста, правне терминологије и индустријског жаргона. То их чини тешким за тумачење [3]. Неструктурисана природа наратива отежава проналажење конкретних информација или упоредивост између компанија. Применом техника НЛП-а на 10-К извештајима могу се аутоматски извући корисни увиди и обрасци који би другачије тешко били уочени [3].

1.4 Циљ рада

Као одговор на наведене изазове и могућности које пружају савремени NLP алати, овај рад је усмерен на развој и евалуацију финансијског агента заснованог на великом језичком моделу (LLM). Конкретно, користимо LLM опште намене уз конструисање контекста како бисмо омогућили агенту да обрађује сложене финансијске документе. Поред самог агента, уводимо и иновативан приступ процене његове успешности: други LLM служи као судија који оцењује и анализира одговоре агента по критеријумима тачности и квалитета.

Архитектура великих језичких модела

2.1 Неуронске мреже

2.1.1 Неурон

У контексту неуронских мрежа, неурон представља основну јединицу обраде информација [4]. Он симулира рад људског неурона у мозгу. Графички изглед једног неурона је приказан на слици 2.1. Сваки неурон прима улазне сигнале $x_0, x_1 ... x_N$, обрађује их, и издаје излазни сигнал y.

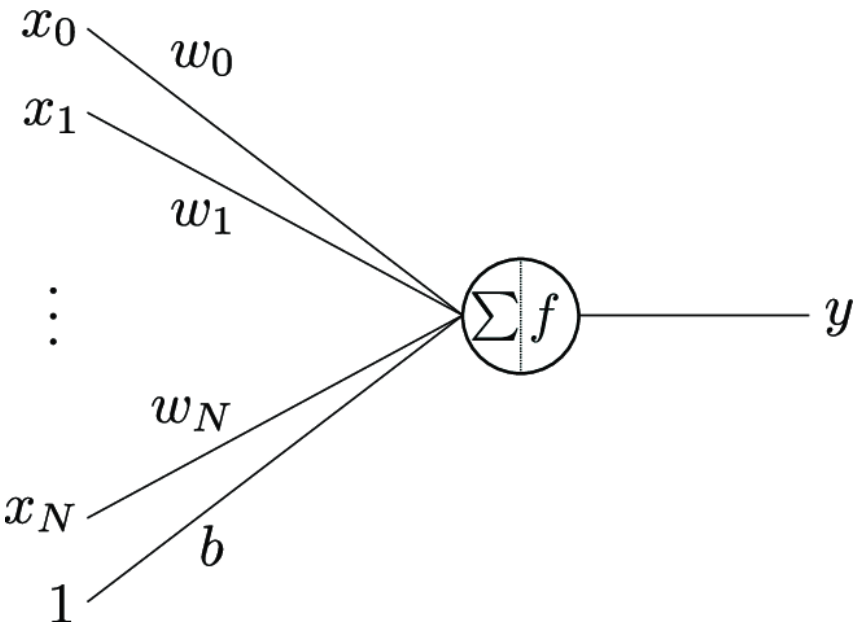
Тежине (енг. *weights*) $w_0, w_1 ... w_N$ представљају параметре који се користе за модификацију улазних података прослеђених неурону. Сваки улаз у неурон је помножен са одговарајућом тежином. Тежине утичу на значајност улазних података и одређују њихову улогу у формирању излаза неурона. Процес учења у неуронским мрежама, познат као обука, састоји се из ажурирања и промене тежина како би мрежа најбоље моделовала жељени задатак.

Збир улаза помножених са тежинама сигнала се прослеђује активационој функцији f. Коришћењем активационих функција, неуронске мреже су у могућности да моделирају нетривијалне односе између улазних и излазних података. Овај излазни сигнал затим служи као улаз за следећи слој неурона у мрежи.

Пристрасност (енг. *bias*) је додатни параметар који се користи за прилагођавање излаза неурона. Тежина којом пристрасност утиче на активациону функцију је одређена тежином b.

Математички израз како тежине, улази и пристрасност утичу на излазни сигнал је дат једначином 2.1.

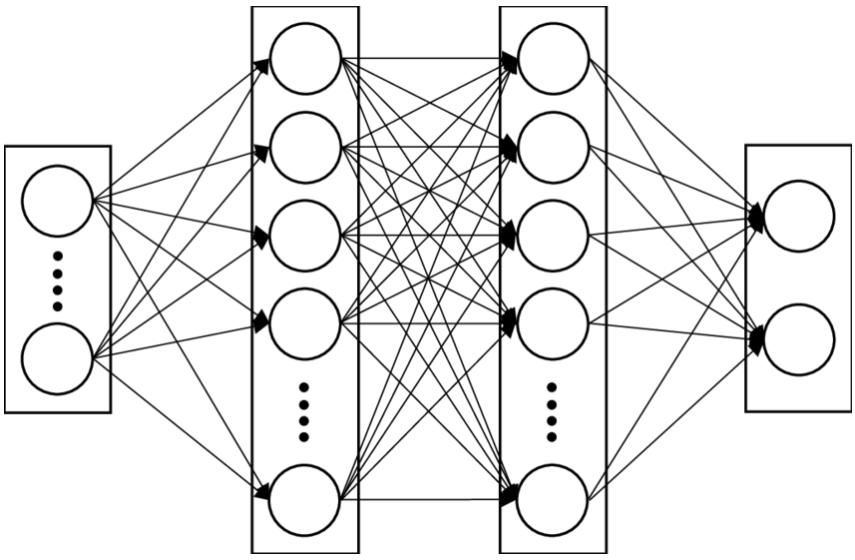
$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.1}$$



Слика 2.1: Графички приказ једног неурона [5]

2.1.2 Потпуно повезане неуронске мреже

Потпуно повезане неуронске мреже (ППНМ) (енг. *Fully Connected Neural Networks*) су скуп повезаних неурона описаних у 2.1.1. Састоји се од више слојева, први слој се назива улазни слој, а последњи слој се назива излазни слој. Сви остали слојеви се називају скривени слојеви (енг. *hidden layer*). На слици 2.2 је графички приказана неуронска мрежа која има 2 скривена слоја. Сваки слој може да има произвољан број неурона, а неуронска мрежа може да има произвољан број скривених слојева. Мењајући архитектуру, односно број неурона у једном слоју и број слојева се мења моћ мреже и комплексност задатка који мрежа може да решава.



Слика 2.2: Графички приказ потпуно повезане неуронске мреже [6]

ППНМ прима само децималне бројеве као улаз и битно је да приликом тренирања и предикције да исте вредности атрибута улазних података улазе у потпуно повезану мрежу на исти улаз. Овакав тип мреже се најбоље показао на проблемима класификације [7].

2.1.3 Софтмакс (*softmax*)

Софтмакс функција се често користи у неуралним мрежама за решавање задатака класификације, као и за генерисање расподеле вероватноћа излаза из мреже. Она трансформише илаз тако да све вредности излаза леже у интервалу између 0 и 1 и да се сума свих вредности излаза једнака 1 Формално, софтмакс функција је дефинисана у изразу 2.2 за вектор излаза Z :

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{2.2}$$

Где:

- z је вектор излаза модела за сваку од N класа.
- е представља експоненцијалну функцију (елемент по елемент).
- i је индекс класе за коју рачунамо вероватноћу.

2.2 Архитектура трансформера

2.2.1 Токенизација

Трансформери раде над дискретним секвенцама токена уместо над сировим текстом, па је први корак токенизација – разлагање текста на атомске јединице (токене) које чине улаз модела. Савремени модели трансформера често користе токенизацију на нивоу подсегмената (eng. *subword*) како би постигли отворени речник, што значи да се свака реч може представити као секвенца подсекмената токена. Распрострањен *subword* метод је Кодирање парова бајтова (енг. Byte Pair Encoding - BPE), алгоритам за компресију података прилагођен раду са текстом. Првобитно га је представио Gage (1994) за компресију, а касније су га на сегментацију речи применили Sennrich и сар. (2016) [8, 9]. Алгоритам итеративно спаја најфреквентнији пар симбола у корпусу, додајући тако насталу целину као нови токен. Понављањем спајања, BPE гради речник уобичајених подсегмената; на пример, ретка реч „nationalism"може се поделити на подсегменте „nation@@"и „alism"(са „@@"као ознака за поделу) на основу фреквенције. Овај процес даје речник фиксне величине састављен од подсегмената које постижу баланс између грануларности на нивоу карактера и холизма на нивоу речи. Теоријска улога

токенизације је да ограничи улазни простор на управљив скуп симбола, а да притом сачува могућност да се од тих симбола конструише било која реч.

2.2.2 Векторска репрезентација токена

После токенизације, сваки токен се трансформише у континуалну векторску репрезентацију (енг. *Embedding*). Теоријска улога векторске репрезентације је да омогући моделу да у наученом векторском простору мери семантичке и синтаксичке сличности између токена. Формално, овај слој се може посматрати као табела претраге – нпр. матрица $\mathbf{E} \in \mathbb{R}^{|V| \times d}$, где је $|V|$ величина речника, а d димензија скривеног слоја модела. Сваки токен t_i мапира се на d-димензионални вектор $\mathbf{x}_i = \mathbf{E}[t_i]$. Ови вектори су параметри који се добијају обучавањем модела, иницијално насумични или претходно тренирани, и оптимизују се током обуке модела да би кодирали корисне лингвистичке информације. Код трансформера, вектори су обично величине d_{model} и скалирају се са $\sqrt{d_{\text{model}}}$ при иницијализацији како би им се величина задржала у разумним границама [10].

Важно обележје корака векторске репрезентације је додавање позиционог кодирања. За разлику од рекурентних мрежа, трансформер нема урођено поимање редоследа речи, па се позиционе информације морају експлицитно увести. Решење које предлажу Vaswani и сар. је да се сваком позиционом векторском репрезентацијом токена дода позициони вектор [10]. Ова позициона кодирања су фиксна и дефинисана су помоћу синусоидалних функција различитих фреквенција [10]. Конкретно, за позицију pos (нумерисану од 0) и индекс димензије i, кодирање је:

$$\text{PE}(pos,\, 2i) = \sin\Big(\frac{pos}{10000^{2i/d_{\text{model}}}}\Big), \qquad \text{PE}(pos,\, 2i+1) = \cos\Big(\frac{pos}{10000^{2i/d_{\text{model}}}}\Big) \tag{2.3}$$

где је d_{model} димензионалност ембединга [10]. Ова наизменична синус-косинус формулација производи позиционе векторе јединствене за сваку позицију и који кодирају релативна растојања. Позиционо кодирање $\mathbf{p}_i$ се додаје вектору токена $\mathbf{x}_i$ како би се добила коначна улазна репрезентација $\mathbf{z}_i = \mathbf{x}_i + \mathbf{p}_i$ која се уводи у трансформер. Због тога модел може да разликује позиције токена и учи односе који зависе од редоследа, а да и даље ради над континуалним векторским репрезентацијама.

2.2.3 Механизам пажње

Према научном раду [10], функција пажње (енг. *Attention head*) се може описати као пресликавање упита и скупа парова кључа и вредности у излазни вектор, при чему су упит, кључеви, вредности и излаз сви вектори. Излаз се израчунава као тежинска сума вектора вредности, а тежина сваке вредности добија се из функције компатибилности

између упита и одговарајућег кључа.

2.2.4 Скалирана пажња заснована на скаларном производу

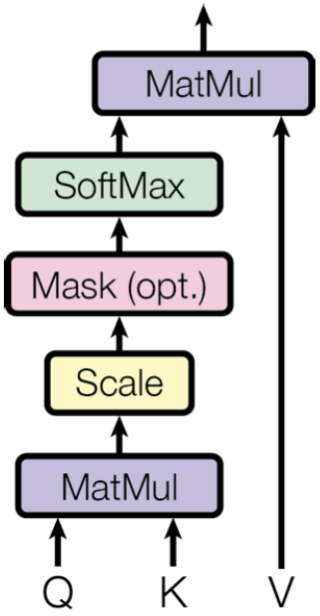
Скалирана пажња заснована на скаларном производу (енг. *Scaled Dot-Product Attention*) је механизам који омогућава моделу да одмери утицај различитих токена при израчунавању репрезентација за следећи слој. У трансформеру, основна јединица је глава пажње са скалираним скаларним производом. Теоријска улога једне главе пажње је да израчуна тежинску комбинацију вектора вредности за сваку позицију, где су тежине одређене паровним сличностима између упита и скупа кључева. Свака глава пажње ради над три скупа вектора: упити (Q), кључеви (K) и вредности (V), димензија d_k, d_k и d_v (често $d_v = d_k$) редом. У интроспективној пажњи (енг. *self-attention*), језгру трансформер слојева, упити, кључеви и вредности долазе из исте секвенце, што омогућава моделу да обрађујући дату позицију „обрати пажњу"на друге позиције у секвенци. Механизам пажње израчунава меру компатибилности између сваког упита и сваког кључа помоћу скаларног производа $Q \cdot K^T$ [10]. Резултат се затим скалира са $\frac{1}{\sqrt{d_k}}$ и нормализују *softmax*-ом како би се добиле тежине пажње. Излаз главе пажње је тежинска сума вектора вредности, користећи те нормализоване тежине. Математички, за скуп Q, K и V, глава пажње даје:

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{Q\,K^T}{\sqrt{d_k}}\Big)\,V \tag{2.4}$$

како су увели Vaswani и сар. [10]. Сваки ред матрице $\text{softmax}(\frac{QK^T}{\sqrt{d_k}})$ представља расподелу вероватноће над свим кључевима за одређени упит, показујући колико пажње (значаја) упит поклања вредности сваког кључа. Добијена тежинска сума даје контекстни вектор за сваки упит, тј. излаз пажње који кодира информације агрегиране из свих позиција, пристрасно у корист оних релевантних за позицију упита.

Скалирање са $\frac{1}{\sqrt{d_k}}$ је кључан теоријски детаљ. Без њега, скаларни производи QK^T расту по величини са већим d_k, што може да доведе тога да *softmax* обрати пажњу на само један токен игноришући остале. Скалирањем скаларних производа инверзним квадратним кореном димензије кључа, вредности које улазе у *softmax* остају умерене чак и кад d_k расте, што емпиријски води стабилнијем учењу [10, 11]. У пракси, употреба скалиране пажње са скаларним производом у трансформеру обезбедила је једноставнију и бржу имплементацију пажње без жртвовања перформанси [10]. Свака глава пажње стога излази секвенцу вектора (по један по улазној позицији) који мешају информације са свих позиција, фокусирајући се на оне процењене као релевантне датом упиту.

Архитектура механизма пажње је приказама на слици 2.3.



Слика 2.3: Архитектура скалиране пажње засноване на скаларном производу

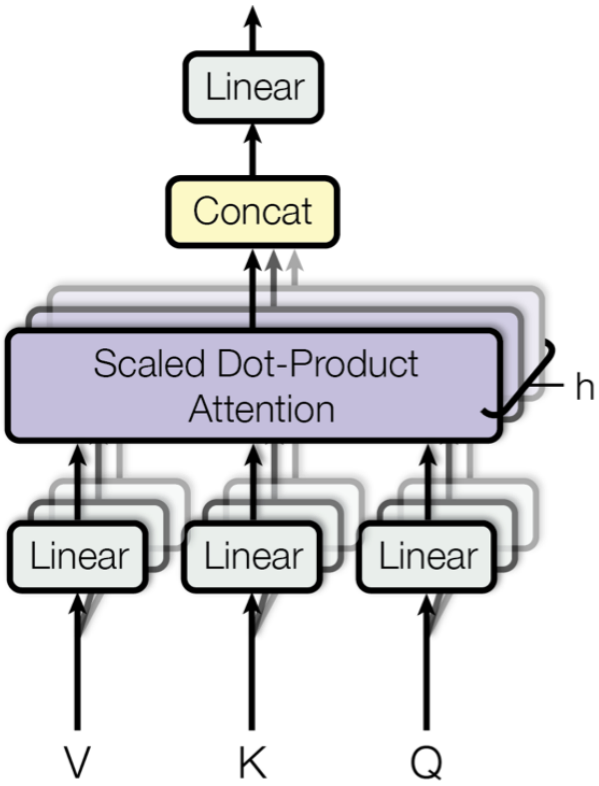
2.2.5 Механизам пажње са више глава

Иако једна глава пажње може да извуче један скуп односа преко секвенце, трансформер користи механизам пажње са више глава (енг. *Multi-Head Attention*) како би моделу омогућио да паралелно обрађа пажњу на више аспеката података. Идеја је да постоји више независних глава пажње (рецимо h глава), свака са сопственим линеарним трансформацијама за упите, кључеве и вредности. Улаз у слој вишеглаве пажње се прво пројектује у h различитих подпростора помоћу h научених линеарних пројекција: за сваку главу i имамо матрице пројекција W_i^Q, W_i^K, W_i^V које мапирају оригиналне d_{model}-димензионалне упите, кључеве и вредности у d_k-димензионе Q_i, K_i, V_i. Типично се бира $d_k = d_{\text{model}}/h$ тако да је укупна рачунања преко h глава упоредива са једном „великом"главом по димензионалности. Свака глава i затим изводи скалирану пажњу са скаларним производом у свом пројектованом подпростору, дајући излазну матрицу $\text{head}_i = \text{Attention}(Q_i, K_i, V_i)$ димензије $n \times d_v$ (за n улазних позиција). Излази h глава се конкатенирају (дуж димензије карактеристика), па се кроз завршну линеарну пројекцију W^O (облика $h \cdot d_v$ на d_{model}) поново комбинују информације. У формулном облику, ако i-ти излаз главе означимо као head_i, излаз вишеглаве пажње је:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)\, W^O \tag{2.5}$$

где је $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ за $i = 1, \ldots, h$ [10]. Ова архитектура (слика 2.4) ефективно покреће h одвојених слојева пажње у паралели [10]. Теоријска

предност вишеглаве пажње је у томе што свака глава може да учи да се фокусира на различите обрасце или односе у подацима.



Слика 2.4: Архитектура механизма пажње са више глава

2.2.6 Трансформер

У пуном енкодер–декодер Трансформеру (енг. *Transformer*), изворна секвенца се најпре токенизује, ембедује и проширује позиционим кодирањем, након чега пролази кроз стек енкодера од N идентичних слојева. Сваки слој енкодера примењује:

1. механизам интроспективне пажње (*self-attention*) која омогућава свакој позицији да обраћа пажњу на све остале у извору
2. позиционо-локални, по елементима, потпуно повезан (*feed-forward*) слој

оба подслеја су обавијена резидуалним („Add") везама и нормализацијом слоја („Norm"). Активности вршног слоја енкодера H представљају контекстом богате репрезентације које делују као меморија са адресирањем по садржају за декодер. Декодер конзумира на десно померену циљну секвенцу са сопственом векторском репрезентацијом и позиционим кодирањем. Сваки слој декодера садржи:

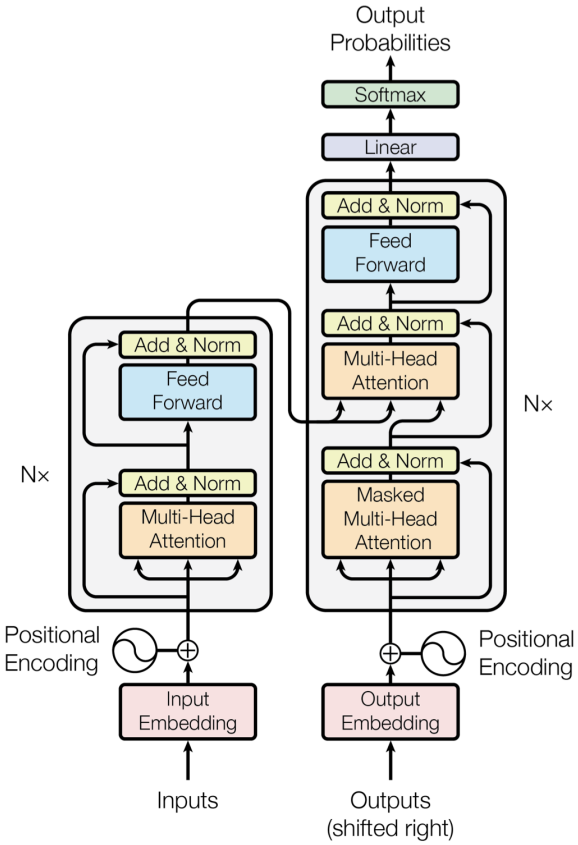
1. маскирану самопажњу, са каузалном (троугластом) маском тако да позиција t не може да види токене $> t$

2. енкодер–декодер *cross-attention*, где упити декодера претражују меморију енкодера N (кључеве/вредности), омогућавајући ослањање на извор

3. *feed-forward* мрежу

и овде резидуалне путање и нормализација слоја стабилизују оптимизацију и очувавају сигнал. Слагање слојева даје хијерархијску композицију: нижи слојеви хватају локалне синтаксичке сигнале, док виши кодирају семантичке односе, при чему механизам пажње са више глава расподељује ове улоге по главама. Коначна стања декодера пролазе кроз линеарну пројекцију и *softmax* (енг. *softmax*) ради добијања вероватноћа наредног токена.

Графички приказ архитектуре трансформера је приказана на слици 2.5.



Слика 2.5: Архитектура трансформера

2.3 Архитектура великог језичког модела

Велики језички модели (ВЈМ) (енг. *Large Language Models - LLM*) заснивају се на трансформер архитектури [10]. ВЈМ је веома дубока хрпа трансформер слојева, при чему сваки слој садржи вишеглаву пажњу и *feed-forward* подслојеве. Ова архитектура омогућава

моделу да одмерава релевантност сваке речи (токена) у улазу у односу на све друге речи, хватајући далекосежне зависности у тексту. Пошто трансформер обрађује речи паралелно, може да се скалира на веома велике величине модела и ефикасно рукује дугим секвенцама [10].

2.3.1 Врсте великих језичких модела

Иако основни трансформер блок остаје језгро, ВЈМ-ови типично имају огроман број параметара распоређених преко многих слојева. На пример, модел GPT-3 компаније OpenAI садржи 175 милијарди параметара и користи 96 трансформер слојева у конфигурацији само-декодера [12]. Сваки слој у GPT-3 има бројне главе пажње (96 глава по слоју) које раде паралелно, што омогућава моделу да прати различите аспекте улазног текста [12].

Упркос својој величини, архитектура већине ВЈМ-а може се категоризовати у неколико основних типова:

- **само-декодерски модели** (попут серије GPT и Meta-иног LLaMA) који генеришу текст предвиђајући следећи токен
- **само-енкодерски модели** (попут BERT-а) намењени задацима разумевања и анализе
- **енкодер–декодер модели** (попут T5) погодни за секвенца-у-секвенцу задатке (превођење, сажимање итд.) [10, 13]

Општенаменски ВЈМ који се користе у чет-ботовима и креативном генерисању текста обично су трансформери само-декодерског типа, претходно тренирани да настављају текст. У свим случајевима, међутим, оквир самопажње трансформера је кичма која омогућава овим моделима да из података науче сложене језичке обрасце и семантику [10]. Резултат је архитектура која, када се скалира, може да испољи изненађујуће богате способности разумевања и генерисања језика.

2.3.2 Ограничења великих језичких модела

Однос између дужине контекста и механизма пажње трансформера је директан. Механизам интроспективне пажње омогућава сваком токену да обрати пажњу на све друге токене у улазу, што је начин на који модел интегрише контекст. Али то има цену: пажња има квадратну сложеност у односу на дужину секвенце. Другим речима, удвостручавање прозора контекста може да учетворостручи потребно рачунање за пажњу [10, 13]. Зато је дужина контекста дуго имала практична ограничења – обрада веома дугих секвенци је спора и захтева много меморије. Тренутни ВЈМ-и се претежно тренирају на релативно кратким исечцима текста (нпр. неколико хиљада токена), што такође значи да можда природно не уче зависности у веома дугим текстовима [13].

2.4 Контекст у великим језичким моделима

У ВЈМ-а, контекст се односи на улазни текст који се моделу пружа и на који се модел условљава да би генерисао одговор. То чини ефективну радну меморију модела, ограничену прозором контекста одређене дужине мерене у токенима, што ограничава колико текста модел може одједном да разматра [**?**]. Све инструкције задатка, позадинске информације и историја конверзације морају бити кодиране у овом контексту, јер LLM-ови током инференције не уче активно нове информације – уместо тога, генеришу излазе искључиво на основу образаца у датом упиту и својих тренираних параметара [**?**]. Другим речима, све што модел треба да зна или да уради за дати упит мора бити обезбеђено у улазном контексту у тренутку извршавања. Већи прозор контекста зато омогућава да се укључи више информација или дужи дијалог, помажући моделу да током дугих интеракција производи кохерентне и релевантне одговоре [**?**]. Међутим, преоптерећивање контекста има мане: повећава рачунање и трошак, а модели могу тешко да уоче релевантне детаље ако је промпт предугачак или има велики шум [**?**]. Истраживања показују да ВЈМ-и често испољавају пристрасности првенства и свежине – теже да се фокусирају на информације на почетку или крају прозора контекста више него на оне у средини [**?**]. Ово сугерише да редослед и позиционирање садржаја у промпту могу утицати на перформансе модела, што је важна напомена при изради ефективних упита [**?**].

2.4.1 Манипулисање контекстом путем изградње упита

Пошто је понашање ВЈМ-а у потпуности вођено улазним текстом, могуће је манипулисање контекстом како би се модел усмерио ка различитим задацима и одговорима. Ова пракса је позната као промпт инжењеринг (енг. *Prompt engineering*): формулисање правих инструкција или примера у промпту да би се изазвао жељени излаз модела. Уместо ажурирања параметара модела, промпт инжењеринг „програмира“ модел природним језиком. Промптови могу бити једноставни – инструкција или питање – или сложени, са структурираним уносом који садржи више примера и ограничења. На пример, може се испред текста додати инструкција као што је „Преведи следећи текст на француски:“ или „Сажми кључне тачке из овог чланка.“ да би се задатак усмерио [**?**]. Ова способност извођења задатака по примеру у контексту, без додатног тренирања модела, обележје је модерних ВЈМ-а и често се назива учење у контексту (енг. *in-context learning*) [**?**].

Технике манипулисања контекстом укључују:

- **Zero-Shot промптовање:** Пружање само инструкције или питања, ослањајући се на стечено знање модела за одговор (нпр. „Објасни зашто је небо плаво.“) [**?**].

- **Few-Shot промптовање:** Давање неколико примера питање–одговор или демонстрација пре стварног упита, да би се моделу показало како да одговара. Ово помаже да се активирају релевантни обрасци из предтренинга путем примера [**?**].
- **Chain-of-Thought промптовање:** Инструкција моделу да резонује кроз проблем корак–по–корак (често додавањем „Хајде да размишљамо корак по корак“) ради побољшања тачности у сложеном резоновању [**?**].
- **Промптовање улогом:** Додавање контекста који уоквирује ко је модел или стил одговора (нпр.: „Ти си стручни медицински асистент. Одговори на питање уз клиничке доказе.“) [**?**].

Стратешким обликовањем контекста овим методама, из истог модела могу се откључати широки опсези способности – од писања кода до одговарања на финансијска питања – без промене самог модела [**?**].

2.4.2 Генерисање са допунским преузимањем

Иако се промпт дизајном може боље искористити оно што ВЈМ већ зна, постоје ситуације када су потребне информације које модел не зна. Генерисање са допунским преузимањем (енг. *Retrieval–Augmented Generation* – RAG) је техника која ово ограничење превазилази увођењем спољног извора знања у контекст који се даје моделу [**?**]. У овом приступу, систем најпре шаље упит ка бази знања или корпусу докумената да дохвати релевантне пасусе, а затим проширује промпт тим преузетим пасусима као додатним контекстом [**?**]. LLM се затим условљава овим обогаћеним контекстом да би генерисао одговор [**?**]. То ефективно опрема модел динамичком, непараметарском меморијом: уместо да се ослања само на оно што је у његовим фиксним параметрима, модел може да користи ажурне информације довучене у тренутку упита [**?**].

Оваквим проширивањем контекста може значајно да се побољша учинак на задацима који захтевају знање. На пример, показано је да је RAG модел достигао добре резултате на бенчмарцима за отворене и доменске задатке са структуром питање–одговор тако што је за сваки упит преузимао релевантне исечке са Википедије [**?**, **?**]. Поред тога, ажурирање знања модела више не захтева скупо поновно тренирање – довољно је освежити или проширити спољну базу знања, а механизам за преузимање ће довести нове информације у контекст модела [**?**].

Важно је уочити однос између генерисања са допунским преузимањем и величине прозора контекста ВЈМ-а [**?**]. Када би се, хипотетички, цела база знања или многи документи могли сместити у промпт, модел би у теорији могао директно да приступи свим тим информацијама без засебног корака преузимања [**?**]. Ипак, постоје практични изазови: веома велики контексти носе велике трошкове и могу довести до превеликог

шума у контексту [**?**, **?**]. Стога ова техника постаје веома релевантна, нарочито за упите који захтевају прецизно издвајање мале количине релевантног знања из огромног корпуса или за праћење најновијих информација као што је случај са финансијским извештајима.

Литература

[1] K. Rocha *et al.*, "Discovering the sentiment in finance's unstructured data," 2021.

[2] Paro AI, "The strategic benefits of NLP in finance," 2023.

[3] Z. Yang *et al.*, "Evaluating LLMs in financial NLP: A comparative study on financial report analysis," 2025.

[4] "Artificial neuron," Aug. 2023. Page Version ID: 1170144985.

[5] Y. Ioannou, *Structural Priors in Deep Neural Networks*. PhD thesis, Sept. 2017.

[6] I. Anwar and N. Ul Islam, "Learned Features are Better for Ethnicity Classification," *Cybernetics and Information Technologies*, vol. 17, Sept. 2017.

[7] "Artificial neural network," Sept. 2023. Page Version ID: 1175471829.

[8] P. Gage, "A new algorithm for data compression," *C Users Journal*, vol. 12, no. 2, pp. 24–35, 1994.

[9] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Association for Computational Linguistics, 2016.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, pp. 5998–6008, Curran Associates, Inc., 2017.

[11] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2015.

[12] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33*, pp. 1877–1901, Curran Associates, Inc., 2020.

[13] Z. Lu, "Large language models: Development in model scale and challenges," *Applied and Computational Engineering*, vol. 114, pp. 154–161, 2024.

[14] K. Martineau, "What's an LLM context window and why is it getting larger?," 2024.

[15] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," 2023.

[16] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," 2025.

[17] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems 33*, pp. 9459–9474, Curran Associates, Inc., 2020.

[18] Q. Yang *et al.*, "Dual retrieving and ranking medical LLM with retrieval augmented generation," *Scientific Reports*, vol. 15, p. 18062, 2025.

[19] M. Kim, "Large context windows versus RAG," 2024.