УНИВЕРЗИТЕТ У БЕОГРАДУ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Евалуација финансијског агента користећи велики језички модел као судију

Мастер рад

Ментор
проф. др Милош Цветановић

Студент
Алекса Рачић, 2023/3001

Београд, септембар 2025

Садржај

Увод

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 овом количином информација постало је озбиљан изазов: огроман број дневних финансијских вести и извештаја једноставно није могуће 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.

Напредак у машинском учењу, а посебно у обради природног језика (енг. *Natural Language Processing - NLP*), драматично је променио начин на који финансијска индустрија обрађује текстуалне податке. Ова обрада 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. Алати обраде природног језика могу брзо да обраде велике количине текста како би уочили трендове, измерили сентимент и истакли потенцијалне ризике, суштински претварајући људски језик у примењива сазнања [2]. Кроз анализу великих текстуалних корпуса, савремени

системи помажу у подршци пословним процесима, откривању макроекономских сигнала и побољшању доношења одлука у финансијским институцијама [3].

Велики скок у способностима обраде природног језика донели су велики језички модели - ВЈМ (енг. *Large Language models - 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 система који су често захтевали специфично дотренирање за сваки задатак, велики језички модели 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.

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. Конкретно, користи се велики језички модел опште намене уз конструисање контекста како би се омогућило агенту да обрађује сложене финансијске документе. Поред самог агента, уводен и иновативан приступ процене његове успешности: други LLM служи као судија који оцењује и анализира одговоре агента по критеријумима тачности и квалитета.

Такође ће се разматрати и значај проширеног контекста за сам квалитет одговора. Уз помоћ агента-судије, биће систематски анализирано у којој мери проширење контекстуалних информација утиче на тачност, потпуност и квалитет одговора на сложена финансијска питања. На овај начин, добија се увид не само у ефикасност предложеног финансијског агента, већ и у општу применљивост техника рада са великим језичким моделима у специјализованом домену.

Теоријски преглед

Ово поглавље пружа систематичан преглед теоријских концепата неопходних за разумевање касније имплементације агента. Најпре се уводе основни елементи вештачких неуронских мрежа, потпуно повезане архитектуре и софтмакс функција као стандардни класификациони излаз. Затим се описује архитектура трансформера са постепеним разлагањем на токенизацију, векторске репрезентације, позиционо кодирање и механизме пажње. Након тога се гради мост ка великим језичким моделима, класификују се њихови типови и разматрају главни структурни и оперативни лимити. Коначно, анализира се концепт контекста у LLM-овима, технике манипулисања упитима, RAG приступ, агентски мод и стратегије резоновања.

2.1 Неуронске мреже

Одељак уводи градивне блокове класичних неуронских архитектура. Најпре се дефинише модел појединачног неурона и његова математичка формулација са тежинама, пристрасношћу и активационом функцијом, затим се прелази на структуру потпуно повезаних мрежа.

2.1.1 Неурон

У контексту неуронских мрежа, неурон представља основну јединицу обраде информација [4]. Он симулира рад људског неурона у мозгу. 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.

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

2.1.2 Потпуно повезане неуронске мреже

Потпуно повезане неуронске мреже (ППНМ) (енг. *Fully Connected Neural Networks*) су скуп повезаних неурона. 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*). 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.

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 [5].

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.
- e представља експоненцијалну функцију (елемент по елемент).
- i је индекс класе за коју рачунамо вероватноћу.

2.2 Архитектура трансформера

Одељак разлаже трансформер на његове концептуалне компоненте: креирање токена, мапирање у континуални простор и механизме пажње. Поступно се гради од основне скалиране пажње засноване на скаларном производу, преко вишеглаве варијанте до комплетне енкодер–декодер архитектуре.

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 за компресију, а касније су га на сегментацију речи применили Sennrich и сарадници [6, 7]. 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 [8].

Важно обележје корака векторске репрезентације је додавање позиционог кодирања. За разлику од рекурентних мрежа, трансформер нема уређено поимање редоследа речи, па се позиционе информације морају експлицитно увести. Решење које предлажу Vaswani и сарадници је да се сваком позиционом векторском репрезентацијом токена дода позициони вектор [8]. Ова позициона кодирања су фиксна и дефинисана су помоћу синусоидалних функција различитих фреквенција [8]. 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 [8]. 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 [8], 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

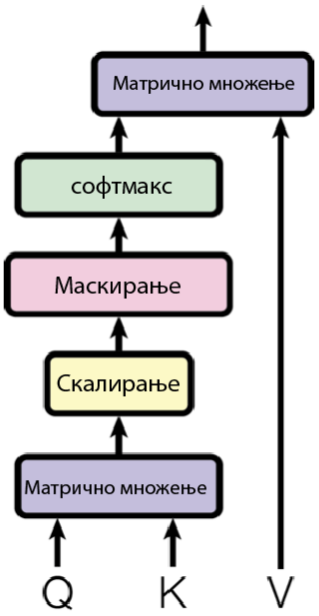
Скалирана пажња заснована на скаларном производу (енг. *Scaled Dot-Product Attention*) је механизам који омогућава моделу да одмери утицај различитих токена при израчунавању репрезентација за следећи слој. У трансформеру, основна јединица је глава пажње са скалираним скаларним производом. Теоријска улога 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 $Q \cdot K^T$ [8]. 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 сарадници [8]. 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 [8, 9]. У пракси, употреба скалиране пажње са скаларним производом у трансформеру обезбедила је једноставнију и бржу имплементацију пажње без жртвовања перформанси [8]. 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 2.1.



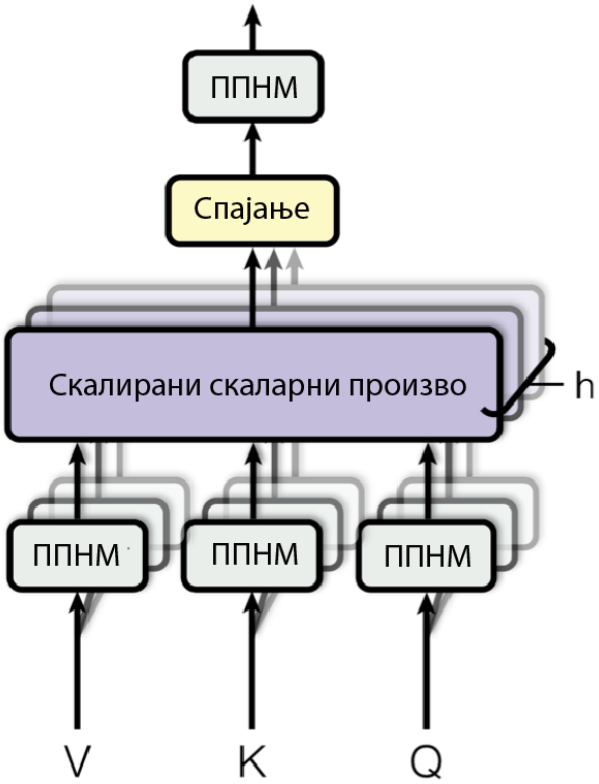
Слика 2.1: Архитектура скалиране пажње засноване на скаларном производу

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, па се кроз завршну линеарну пројекцију W^O поново комбинују информације. У облику формула, ако i-ти излаз главе означимо као head_i, излаз вишеглаве пажње је приказан у једначини 2.5:

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

где је $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ за $i = 1, \ldots, h$ [8]. Ова архитектура (слика 2.2) ефективно покреће h одвојених слојева пажње у паралели [8]. Теоријска предност вишеглаве пажње је у томе што свака глава може да учи да се фокусира на различите обрасце или односе у подацима.



Слика 2.2: 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, слој потпуно повезане неуронске мреже

Оба подслеја су обавијена резидуалним (енг. *Add*) везама и нормализацијом слоја (енг. *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 интроспективну пажњу, са каузалном маском тако да позиција t не може да види токене $> t$
2. енкодер-декодер (енг. *cross-attention*), где упити декодера претражују меморију енкодера N (кључеве/вредности), омогућавајући ослањање на извор
3. потпуно повезану неуронску мрежу

Додатно, резидуалне везе 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 софтмакс функцију ради добијања вероватноћа наредног токена.

Графички приказ архитектуре трансформера је приказана на слици 2.3.

2.3 Архитектура великог језичког модела

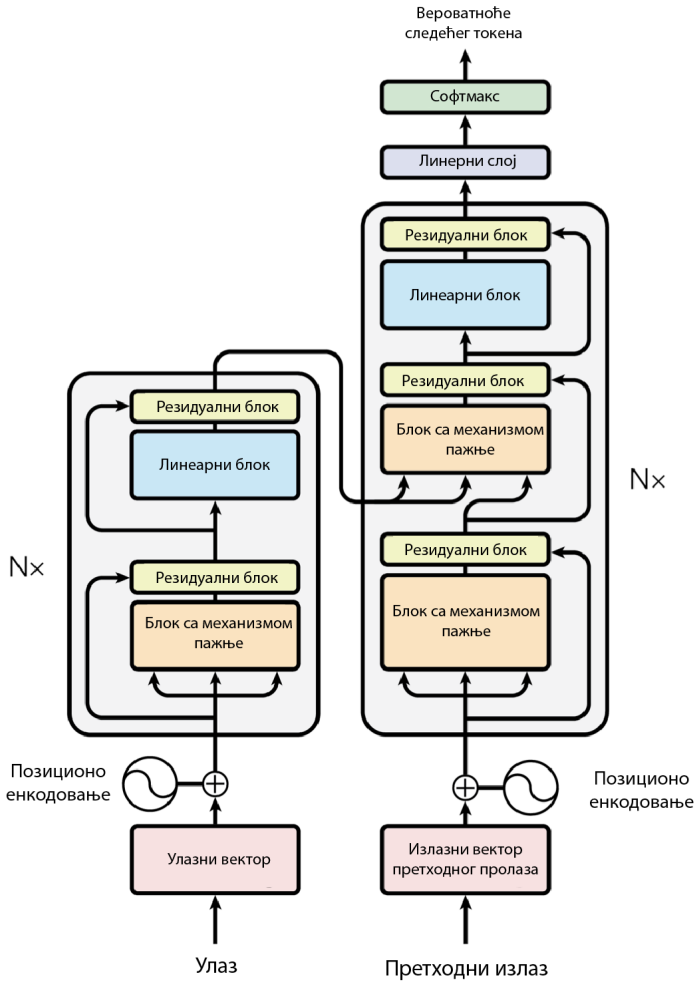
Велики језички модели (LLM) (енг. *Large Language Models - LLM*) заснивају се на трансформер архитектури [8]. LLM је веома дубока хрпа трансформер слојева, при чему сваки слој садржи вишеглаву пажњу и *feed-forward* подслојеве. Ова архитектура омогућава моделу да одмерава релевантност сваке речи у улазу у односу на све друге речи, хватајући зависности у тексту. Пошто трансформер обрађује речи паралелно, може да се скалира на веома велике величине модела и ефикасно рукује дугим секвенцама [8].

2.3.1 Врсте великих језичких модела

Иако основни трансформер блок остаје језгро, LLM-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 [10]. Сваки слој GPT-3 садржи 96 бројне главе пажње које раде паралелно, што омогућава моделу да прати различите аспекте улазног текста [10].

Архитектура већине LLM-а може се категоризовати у неколико основних типова:

- **декодерски модели** (попут серије GPT и Meta-иног LLaMA) који генеришу текст предвиђајући следећи токен
- **енкодерски модели** (попут BERT-а) намењени задацима разумевања и анализе



Слика 2.3: Архитектура трансформера

- **енкодер-декодер модели** (попут T5) погодни за превођење и сажимање [8, 11]

Општенаменски LLM који се користе у чет-ботовима и креативном генерисању текста обично су трансформери декодерског типа, претходно тренирани да настављају текст. У свим случајевима, међутим, механизам интроспективне пажње трансформера је срж која омогућава овим моделима да из података науче сложене језичке обрасце и семантику [8]. Резултат је архитектура која, када се скалира, може да испољи изненађујуће способности разумевања и генерисања текста.

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, удвостручава-

ње прозора контекста може да учетворостручи потребно рачунање за пажњу [8, 11]. Зато је дужина контекста дуго имала практична ограничења. Обрада веома дугих секвенци је спора и захтева много меморије. Тренутни LLM-и се претежно тренирају на релативно кратким исечцима текста, што такође значи да можда природно не уче зависности у веома дугим текстовима [11].

2.4 Контекст у великим језичким моделима

У LLM-а, контекст се односи на улазни текст који се моделу пружа и на који условљава начин на који модел генерише одговор. То чини ефективну радну меморију модела, ограничену прозором контекста одређене дужине мерене у токенима, што ограничава колико текста модел може одједном да разматра [12]. Све инструкције задатка, позадинске информације и историја конверзације морају бити у овом контексту, јер LLM-ови током инференције не уче активно нове информације, већ уместо тога, генеришу излазе искључиво на основу образаца у датом упиту и својих тренираних параметара [12]. 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 [12]. Међутим, преоптерећивање контекста има мане: повећава рачунање и трошак, а модели могу тешко да уоче релевантне детаље ако је упит предугачак или има велики шум [13]. Истраживања показују да LLM-и често испољавају пристрасности првенства и свежине, односно настоје да се фокусирају на информације на почетку или крају прозора контекста више него на оне у средини [13]. Ово сугерише да редослед и позиционирање садржаја у промпту могу утицати на перформансе модела, што је важна напомена при изради ефективних упита [13].

2.4.1 Манипулисање контекстом путем изградње упита

Пошто је понашање LLM-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 [14]. Ова способност извођења задатака по примеру у контексту, без додатног тренирања модела,

обележје је модерних LLM-а и често се назива учење у контексту (енг. *in-context learning*) [14].

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.") [14].
- **Few-Shot промптовање:** Давање неколико примера питања и одговора или, да би се моделу показало како да одговара. Ово помаже да се активирају релевантни обрасци из предтренинга путем примера [14].
- **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 [14].
- **Промптовање улогом:** Додавање контекста који уоквирује ко је модел или стил одговора (нпр.: „Ти си стручни медицински асистент. Одговори на питање уз клиничке доказе.") [14].

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 [14].

2.4.2 Генерисање са допунским преузимањем

Иако се промпт дизајном може боље искористити оно што LLM 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 [15]. 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 [15]. LLM се затим условљава овим обогаћеним контекстом да би генерисао одговор [15]. 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 [15].

Оваквим проширивањем контекста може значајно да се побољша учинак на задацима који захтевају знање. На пример, показано је да је RAG модел потигао добре резултате на бенчмарцима за отворене и доменске задатке са структуром питање–одговор тако што је за сваки упит преузимао релевантне исечке са Википедије [15, 16]. 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 [15].

Важно је уочити однос између генерисања са допунским преузимањем и величине прозора контекста LLM-а [17]. 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 корака преузимања [17]. Ипак, постоје практични изазови: веома велики контексти носе велике трошкове и могу довести до превеликог шума у контексту [13, 17]. 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.

2.4.3 Велики језички модел као агент

Манипулисањем контекста могуће LLM претворити у агента који може да доноси одлуке и делује у сложеном окружењу. У овом контексту, агент је систем који користи контекст LLM-а како би у наставки генерисања одговора донео неку одлуку. У контексту LLM-а се описује окружење које агент посматра, укључујући и његове могућности деловања. Агент може да користи LLM за резоновање о окружењу, планирање корака и доношење одлука на основу текућег стања. На пример, у систему за питање-одговор, агент може да одлучи који алат или функцију треба да позове (нпр. претрага базе података, израчунавање статистике) на основу упита корисника и доступних ресурса. Овај приступ омогућава LLM-у да превазилази ограничења статичког знања тако што може динамички да прибавља информације или обавља прорачуне током интеракције.

Приликом осмишљавања логике резоновања агента, издвајају два приступа како БЈМ доноси одлуке о коришћењу алата током решавања задатка:

1. ”Испланирај и уради” приступ (енг. *plan and execute*)
2. Реактивни приступ (енг. *reactive approach*)

”Испланирај и уради” приступ

У овом приступу, агент најпре покушава да изради целовит план корака пре него што започне извршење. То значи да LLM у оквиру једног упита генерише секвенцу акција (нпр. којим редом и које алате треба позвати) како би дошао до решења, а потом се ти кораци спроводе један по један. Предност оваквог начина је што може бити користан код веома сложених задатака који захтевају дугорочно планирање. Међутим, ово долази по цену већег когнитивног оптерећења модела и потенцијалних грешака у планирању.

Ако LLM не успе тачно да предвиди све што је потребно, унапред сачињен план може бити непотпун или погрешан, што доводи до тога да агент креће у погрешном смеру. У контексту нашег финансијског домена, где су питања углавном директно усмерена на конкретне податке или прорачуне, овако детаљно планирање у већини случајева није неопходно.

Реактивни приступ

Реактивни приступ подразумева да агент размишља и дела у наизменичним корацима, доносећи одлуке о наредној акцији у ходу, на основу тренутно доступних информација. Овај стил резоновања често се назива ReAct (енг. **Re**asoning + **Ac**ting). У пракси, то значи да LLM анализира кориснички упит (или међурезултат ако је неки корак већ обављен), одлучује који алат у том тренутку треба применити, извршава тај алат, па затим сагледава добијени резултат и одлучује о следећем кораку. Агент се, дакле, прилагођава у реалном времену без унапред фиксног плана. Предност реактивног приступа је његова једноставност и флексибилност: модел може да реагује на непредвиђене околности или новодобијене информације тако што ће променити ток акција у ходу. Такође, избегава се двоструко ангажовање модела (посебно одвојена фаза планирања па извршавања), што смањује укупан когнитивни терет и време извршења за типичне задатке.

Мана овог приступа јесте потенцијална бесконачна петља која може да настане. Агент у сваком кораку тражи примењује исту акцију, који враћа исти резултат на осноу које агент опет ради исту акцију. Због тога је битно направити механизам прекида уколико се током извршења примењује велики број акција.

У контексту финансијских извештаја, могућност агентског размишања LLM се може искористити као динамичка стратегија на основу које ће се прикупити све релевантне информације како би корисник обио потпун одговор.

Подаци

Поглавље систематизује изворе и структуру података који подупиру евалуацију агента. Најпре се представља EDGAR као примарна регулаторна база, уз нагласак на типовима пријава (10-K, 10-Q, 8-K) које чине основу за извлачење квалитативних и квантитативних финансијских сигнала. Потом се детаљно анализира структура саме форме 10-K кроз њене делове и ставке, чиме се поставља карта садржаја за каснију таргетирану екстракцију. На крају се описује FinanceBench скуп података који служи као стандаризован референтни тест за евалуацију квалитета одговора финансијског агента.

3.1 Преглед базе података EDGAR

Програм електронског прикупљања, анализе и повлачења (енг. *Electronic Data Gathering, Analysis, and Retrieval* - EDGAR) је примарна база података Комисије за хартије од вредности Сједињених Америчких Држава (SEC) за финансијске пријаве. Све јавне компаније дужне су да своје изјаве, периодичне извештаје и друге обрасце за обелодањивање података подносе путем EDGAR-а, уместо папирних пријава. EDGAR ове пријаве чини бесплатно доступним јавности, омогућавајући инвеститорима да преузимају и претражују хиљаде докумената компанија и фондова.

3.1.1 Врсте извештаја доступних на EDGAR-у

EDGAR организује пријаве према стандардизованим типовима образаца. Кључни извештаји доступни у бази EDGAR укључују:

- **Образац 10-K (енг. *Form 10-K*) (Годишњи извештај):** Свеобухватан годишњи извештај који америчке јавне компаније морају да поднесу након завршетка сваке фискалне године. Образац 10-K садржи ревидиране годишње финансијске извештаје, дискусију о пословању компаније и материјалним факторима ризика, као и менаџментову дискусију и анализу финансијских резултата за годину.
- **Образац 10-Q (енг. *Form 10-Q*) (Квартални извештај):** Краћи извештај који се подноси за фискални квартала. Образац 10-Q укључује кварталне финансијске

извештаје, ажурирања о свим значајним променама или материјалним ризицима од последњег 10-K/10-Q.

- **Образац 8-K (енг. *Form 8-K*) (Текући извештај):** Извештај који се подноси ради обелодањивања великих корпоративних догађаја у реалном времену, уместо чекања на наредни 10-Q или 10-K. Компаније подносе 8-K кад год се догоде значајни догађаји које акционари треба да знају.

3.2 Преглед форме годишњег извештаја 10-K

Образац 10-K је годишњи извештај који прописује SEC и који америчке јавне компаније подносе, а нуди детаљну слику финансијског стања и пословних активности компаније током претходне године. 10-K укључује темељан опис пословних операција компаније, дискусију о ризицима са којима се компанија суочава, и ревидиране финансијске резултате за фискалну годину. Менаџмент компаније такође пружа анализу и контекст за финансијске резултате, објашњавајући покретаче перформанси и све трендове или неизвесности који би могли утицати на будуће резултате.

3.2.1 Садржај и структура извештаја Форма 10-K

Форма 10-K је подељена на четири главна дела, при чему сваки део садржи неколико ставки како је прописано регулативом SEC-а. Укупно, постоји 15 нумерисаних ставки које морају бити обухваћене у 10-K. У наставку је преглед ових делова и информација које сваки садржи:

Део I – Преглед пословања и ризика

- **Ставка 1. Пословање (енг. *Item 1. Business*):** опис операција, главни производи/услуге, зависна друштва, тржишта, конкуренција, регулатива, сезоналност.
- **Ставка 1A. Фактори ризика (енг. *Item 1A. Risk Factors*):** најзначајнији ризици по компанију/хартије, набројани по значају.
- **Ставка 1Б. Нерешени коментари особља (енг. *Item 1B. Unresolved Staff Comments*):** материјални нерешени коментари Комисије за хартије од вредности (SEC).
- **Ставка 2. Непокретности (енг. *Item 2. Properties*):** значајна физичка имовина (погони, капацитети, рудници, канцеларије, некретнине).
- **Ставка 3. Судски спорови (енг. *Item 3. Legal Proceedings*):** парнице и судски поступци.

- **Ставка 4. Обелодањивања о безбедности у рудницима (енг. *Item 4. Mine Safety Disclosures*):**

Део II – Финансијске информације и резултати

- **Ставка 5. Тржиште акција и питања акционара (енг. *Item 5*):** берзе, број акционара, политика дивиденди, откупи и емисије.
- **Ставка 6. Одабрани финансијски подаци (енг. *Item 6. Selected Financial Data*):** кључни показатељи (обично последњих 5 година) у сажетку.
- **Ставка 7. Коментари менаџмента (енг. *Item 7. Management's Discussion and Analysis*):** анализа стања и резултата, ликвидност, ресурса, трендова и неизвесности као и кључне процене.
- **Ставка 7А. Тржишни ризик (енг. *Item 7A. Quantitative and Qualitative Disclosures About Market Risk*):** изложености (каматни, девизни, робни, цена капитала) и управљање ризицима.
- **Ставка 8. Финансијски извештаји и додатни подаци (енг. *Item 8*):** Биланс стања, биланс успеха, извештај о токовима готовине и капитал.
- **Ставка 9. Промене и неслагања са рачуновођама (енг. *Item 9*):** промена ревизора и материјална неслагања.
- **Ставка 9А. Контроле и процедуре (енг. *Item 9A. Controls and Procedures*):**
- **Ставка 9Б. Остале информације (енг. *Item 9B. Other Information*):** информације које су требале у Форми 8-К у Q4, а нису објављене.

Део III – Руководство и управљање

- **Ставка 10. Директори, извршни руководиоци и корпоративно управљање (енг. *Item 10*):** биографије, кодекс етике, структура одбора/чланства.
- **Ставка 11. Накнаде извршних руководилаца (енг. *Item 11. Executive Compensation*):** плате, бонуси, акцијске награде, политике програма.
- **Ставка 12. Власништво и питања акционара (енг. *Item 12. Security Ownership*):** власништво >5%, руководиоци/директори; планови капиталних компензација.
- **Ставка 13. Повезана лица и независност директора (енг. *Item 13. Certain Relationships*):** материјалне трансакције са инсајдерима и независност директора.
- **Ставка 14. Накнаде и услуге главног рачуновође (енг. *Item 14. Principal Accountant Fees and Services*):** накнаде ревизорској фирми по врстама услуга.

Део IV – Прилози и финансијски распореди

- **Ставка 15. Прилози, финансијски распореди (енг. *Item 15. Exhibits, Financial Statement Schedules*):** списак свих прилога/распореда (оснивачки акти, статут, значајни уговори, списак зависних, сертификати CEO/CFO).

Сваки од ових делова и ставки организован је на конзистентан начин за све компаније, што олакшава навигацију кроз 10-K.

3.3 Скуп података за тестирање - FinanceBench

FinanceBench је референтни тест за одговарање на финансијска питања. Обухвата укупно 10.231 пар питања и одговора о јавним компанијама у САД. Питања су утемељена на стварним корпоративним извештајима из EDGAR-а описаним у секцији 3.1. Укупно, скуп података покрива 40 компанија из САД у различитим индустријским секторима. Свако питање прати верификован одговор и подржавајући исечак доказа који оправдава одговор. Лабелари и финансијски стручњаци, осигурали су да свако питање буде јасно и једноставно за одговор на основу извештаја, тако да бенчмарк поставља минимални стандард перформанси за тачност модела. Табела 3.1 сумира кључне карактеристике скупа података FinanceBench [18].

3.3.1 Типови питања и композиција

Питања у FinanceBench су категоризована у три групе које одражавају начин на који су генерисана и вештине потребне за одговор:

1. **Доменски релевантна питања:** Фиксни скуп од 25 питања која су широко примењива у анализи било које јавне компаније. Она укључују питања које би поставио финансијски аналитичар, као што су да ли је компанија исплатила дивиденду у последњој години или да ли су оперативне марже остале стабилне током времена.

2. **Ново-генерисана питања:** Ово су оригинална питања која су писали финансијски аналитичари, специфична за контекст сваке компаније, садржај извештаја и индустрију. Нова питања покривају разнолике теме (нпр. пословну стратегију компаније, значајне догађаје или необичне ставке), и формулисана су на различите начине да имитирају упите из стварне праксе. Овај подскуп је примењен на 37 компанија, при чему свака има приближно између 15 и 80 прилагођених питања (просечно 36), укупно 1.323 П-О инстанце.

Карактеристика	**Опис**
Укупно парова П-О	10.231 питање о јавним компанијама, свако са златним одговором и пратећим доказом [18].
Обухваћене компаније	40 јавних компанија (САД), у више индустријских сектора (ГИКС — енг. *Global Industry Classification Standard*, GICS).
Изворни документи	361 финансијски извештај (нпр. 10-K, 10-Q, 8-K, транскрипти позива о заради) из периода 2015–2023.
Подскуп отвореног кода	150 питања и одговора јавно објављених са анотацијама.
Поља по уносу	Питање (финансијски упит), Одговор (анотирани тачан одговор), Доказ (одломак из поднеска компаније са референтном страницом), Образложење (опционално објашњење начина закључивања).
Категорије питања	3 типа: Доменски релевантна (енг. *Domain-Relevant*; општа аналитичка питања), Ново-генерисана (енг. *Novel-Generated*; експертски креирана, специфична за компанију), Метрички генерисана (енг. *Metrics-Generated*; питања о финансијским метрикама).
Потребно расуђивање	Означено типом расуђивања: ∼28% чиста информациона екстракција, 66% укључује нумеричке прорачуне, 6% логичко/аналитичко расуђивање.

Табела 3.1: Карактеристике скупа података FinanceBench

3. **Програмски генерисана питања:** Убедљиво највећи део чине питања аутоматски изведена из финансијских метрика. Анотатори су најпре извукли ∼18 фундаменталних финансијских метрика (нпр. приход, нето добит, различите билансне и новчане ставке) из извештаја сваке компаније током 8 година (2015–2022). Полазећи од њих, скрипта је генерисала бројна питања о базним метрикама и различитим изведеним метрикама рачунатим из њих. На пример, шаблони би формирали питања као што су „Колика је бруто маржа компаније X у 2021?"или „Који је однос амортизације (из извештаја о токовима готовине) према укупном приходу за 2021?", где се одговор може израчунати из пријављених бројки. Нека од ових питања су чисто екстрактивна (траже једну наведну цифру), док друга подразумевају вишестепену аритметику или комбиновање података из више извештаја [18]. Метричка генерација је примењена на 32 компаније, производећи између 135 и 348 питања по компанији (≈249 у просеку), укупно 7.983 пара [18]. Ова категорија тестира способност модела да спроведе нумеричко резоновање.

Преглед алата и радног оквира

Ово поглавље поставља технолошке темеље система: од избора основног LLM модела, преко оквира за оркестрацију, до доменски специфичних библиотека за рад са 10-K извештајима.

4.1 Основни модел

У оквиру развоја агента заснованог на великим језичким моделима, први корак је одабир одговарајућег основног модела. Разматрају се водећи LLM модели у рангу GPT-4, укључујући OpenAI GPT-4, Anthropic Claude 2, DeepSeek и Meta LLaMA 2. Ови модели се упоређују по перформансама, величини, величини контекста, доступности и цени, као и могућностима за позивање алата (енг. tool calling). Нарочито је битно размотрити да ли модел подржава интеграцију са спољним алатима, јер наш агент треба да користи екстерне алате ради проширивања функционалности.

Да бисмо систематично сагледали карактеристике, табела испод пореди ове моделе:

Модел	**Контекст**	**Доступност/API**	**Подршка за алате**
OpenAI GPT-4	8k–32k (до 128k у новијим верзијама)	Комерцијални API	Да[a]
Anthropic Claude 2	100k токена	Комерцијални API	Делимично – преко инструкција (без званичног API-ја за функције)
DeepSeek LLM	до 128k токена	Отворен код	Не – кроз спољну оркестрацију
Meta LLaMA 2 (70B)	∼4k токена	Отворен код	Не – кроз спољну оркестрацију

[a] OpenAI подржава опис интерфејса функција и поврат структурираних JSON позива [19].

Табела 4.1: Поређење напредних LLM модела у рангу GPT-4 по величини, контексту, цени и подршци за алате.

Из табеле је јасно да GPT-4 остаје златни стандард по квалитету генерације и разумевања. Иако је власнички модел затвореног кода, GPT-4 демонстрира врхунске перформансе на бројним задацима: на пример, постиже успех од 86.4% на мултидисциплинарном тесту MMLU у поређењу са $\sim 68.9\%$ код LLaMA 2 [20]. Такође надмашује отворене

моделе у кодирању (HumanEval тест $\sim 67\%$ насупрот $\sim 30\%$ за LLaMA 2) и математичком резоновању. Anthropic Claude 2 је по квалитету често упоредив са GPT-4, али му је главна предност изузетно велики контекст од $100\,000$ токена, што омогућава унос стотина страна текста у једном упиту; притом је цена по 1000 токена око $0.011 за улаз и $0.033 за излаз, што је знатно повољније од GPT-4 [21]. Насупрот томе, отворени модели DeepSeek и LLaMA 2 доносе предност у погледу доступности и трошкова – могу се самостално хостовати без плаћања по упиту. DeepSeek је нарочито занимљив због своје Mixture-of-Experts архитектуре: иако има укупно 671B параметара, за било који задатак активно је само ~ 37B параметара, што смањује рачунарни трошак, а при том постиже висок учинак (нпр. $\sim 73.8\%$ на HumanEval) и подржава дугачак контекст до 128k токена [22]. Међутим, упркос напретку ових модела, OpenAI GPT-4 и даље има предност у општој поузданости и способности сложеног резоновања.

Посебно значајан фактор за изградњу агента је способност модела да позива спољне алате. У том погледу, OpenAI GPT-4 нуди готову подршку за позивање функција. Механизам где се моделу опише интерфејс функције, након чега модел по потреби враћа структуриран JSON позив те функције уместо обичног текста [19]. Ова функционалност омогућава да GPT-4 поуздано иницира извршавање екстерних алата или API позива ради прибављања додатних информација или извршавања задатака. Насупрот томе, други модели у табели тренутно немају уграђену подршку за директно позивање функција. Claude 2 се може инструисати да користи алате путем промпта, али не постоји званичан API за функције те се свака интеграција мора ручно оркестрирати. Слично, LLaMA 2 и DeepSeek, као отворени модели, могу користити алате једино уз помоћ спољних библиотека и логике.

Имајући у виду све наведено, најсавременији квалитет генерисања текста, широку подршку у академским и индустријским тестовима, као и нативну могућност позивања алата – за ову истраживачку примену одабрали смо OpenAI GPT-4 као основни LLM модел. Иако захтева финансијска улагања, GPT-4 ће обезбедити највиши ниво разумевања сложених извештаја и поуздано руковање алатима неопходним за анализу 10-K финансијских докумената. Очекивано је да ће овај модел, уз правилну оркестрацију, најбоље испунити захтеве агента у погледу тачности и функционалности.

4.2 Преглед алата за окетрацију и обраду упита

За изградњу комплексног система заснованог на великом језичком моделу, потребно је користити одговарајуће библиотеке за оркестрацију у раду са великим језичким моделима и инжењеринг упита. Ове библиотеке пружају структуру и алате да се LLM интегрише у апликацију, управља током разговора, памти контекст и позива спољне

изворе података или алате. На располагању је више популарних open-source решења, као и комерцијалних оквира, за ову намену. Нека од значајних су:

- **LangChain** – отворени оквир за оркестрацију LLM апликација написан у Python и JavaScript. Пружа модуларне компоненте за креирање ланаца операција и агената, са подршком за меморију, шаблоне упита и интеграцију алата [23, 24].
- **LlamaIndex** (раније GPT Index) – библиотека фокусирана на повезивање LLM-ова са документима путем индекса (нпр. за претрагу информација у великом корпусу докумената) [24].
- **Haystack** – оквир за претраживање који подржава интеграцију LLM-ова у проточној обради за преузимање информација и одговарање на упите [24].
- **Semantic Kernel** (Microsoft) – SDK за креирање комплексних апликација са LLM-овима, наглашавајући могућност проширења (енг. plugins) и композицију функција у оквиру .NET/Python окружења.

Наведена решења омогућавају оркестрацију позива LLM модела и других услуга у више корака. На пример, уместо да једноставно пошаљемо упит моделу, помоћу ових библиотека можемо раздвојити сложени задатак на секвенцу корака: претрага релевантних докумената, сумирање делова текста, затим генерисање финалног одговора. Оркестрациони оквир управља током извршавања ових корака и разменом података са LLM-ом. Осим тога, библиотеке за LLM оркестрацију апстрахују детаље конкретних API-ја различитих провајдера модела, што омогућава лако експериментисање и замену модела у позадини [23, 24]. На пример, једноставно је могуће конфигурисати систем да користи GPT-4 за један задатак, а Claude 2 или LLaMA за други, без значајних промена у логици апликације.

После разматрања више опција **LangChain** се издваја као најпогоднија библиотека за пројекат. Разлози за одабир LangChain-а произилазе из његових предности:

- **Модуларна архитектура ланаца и агената:** LangChain омогућава дефинисање ланаца (енг. *chains*) – секвенци операција које укључују LLM позиве и друге акције – као и агената који користе LLM за доношење одлука о наредним корацима [24].
- **Интеграција вишеструких LLM модела:** Уграђана подршка за бројне провајдере и моделе, са обједињеним интерфејсом ка OpenAI, Anthropic, HuggingFace и др. [23, 24].

- **Памћење и управљање контекстом:** Механизми меморије за конверзацијске системе (краткорочно и дугорочно) који омогућавају природније дијалоге [24].
- **Шаблонирање и инжењеринг упита:** Шаблони промптова и динамичко попуњавање контекстом за конзистентне улазе моделу [24].
- **Повезивање са спољним алатима и изворима:** Једноставна интеграција претраживача, база знања, калкулатора, веб API-ја, web scraping-а и база података [23, 24].
- **Заједница и документација:** Богата документација и активна заједница, са многим примерима и туторијалима [23, 24].

На основу наведених карактеристика, LangChain се истиче као свеобухватан и флексибилан оквир погодан за наш задатак. Он нам омогућава да дефинишемо агента који корак-по-корак обрађује упит корисника: од иницијалног разумевања питања, преко претраживања потребних информација у 10-K извештајима, до генерисања коначног одговора који комбинује пронађене податке и закључке модела. Алтернативна решења (поменута горе) такође нуде сличне могућности, али LangChain представља стандард у индустрији због богатства функција и доказа успешних примена.

4.2.1 Преглед LangChain библиотеке

LangChain је свеобухватан оквир који омогућава развој сложених апликација заснованих на великим језичким моделима [25]. У наставку се детаљније разматрају кључне компоненте ове библиотеке које су релевантне за имплементацију финансијског агента.

Креирање упита

LangChain користи концепт ланаца (енг. Chain) за извршавање упита над LLM моделима. При томе се најчешће прави упитни шаблон који садржи места за унос променљивих вредности, а затим се тај шаблон повезује са моделом у оквиру ланца. Упитни шаблон (енг. Prompt Template) је предложак текста који комбинује фиксна упутства са променљивим деловима које попуњава кориснички унос [25]. На пример, можемо дефинисати шаблон који преводи задати текст на француски, где је само део текста променљив. Када направимо такав шаблон, креирамо објекат ланца који повезује изабрани LLM модел са датим шаблоном упита. На тај начин, сваки пут када позовемо ланац, довољно је да проследимо конкретне вредности за променљиве у шаблону, а LangChain ће аутоматски саставити пуни упит и проследити га моделу [25].

Кориснички унос се дакле убацује у ипитни шаблон и формира се коначан текст упита, који LLM обрађује и враћа резултат. Следи пример израде једноставног ланца са prompt шаблоном и његовог покретања:

```
from langchain import PromptTemplate, LLMChain
from langchain.chat_models import ChatOpenAI

# 1. Дефинишемо prompt шаблон са променљивом {text}
prompt = PromptTemplate(
    template="Преведи на француски следећи текст: {text}",
    input_variables=["text"]
)

# 2. Иницијализујемо LLM модел (нпр. OpenAI ChatGPT модел)
llm = ChatOpenAI(model_name="gpt-4o", temperature=0)

# 3. Креирамо ланац који повезује модел и prompt шаблон
chain = LLMChain(llm=llm, prompt=prompt)

# 4. Покрећемо ланац прослеђивањем конкретне вредности за {text}
result = chain.run({"text": "Hello, how are you?"})
print(result)
```

Структура 4.1: Пример креирања ланца са prompt шаблоном

У примеру кода 4.1, упитни шаблон садржи текст са уметнутом променљивом *{text}* и дефинише се листа његових променљивих. Затим се иницијализује LLM и креира се LLMChain који инкапсулира логику: узима задати prompt шаблон, попуњава га уносом, шаље моделу упит и враћа одговор. Позив *chain.run* аутоматски убацује дат текст у место *{text}* и прослеђује комплетан упит моделу, након чега добијамо преведени текст као резултат. Оваква структура омогућава да једном дефинишемо шаблон и модел, а да затим више пута извршавамо упите само прослеђивањем различитих улазних вредности.

Регистрација алата

У изградњи агената, алати (енг. Tools) представљају функције или оперативне методе које агент може динамички позивати током свог резоновања. Сваки алат се састоји од саме функције коју позива LLM, али и од метаподатака: пре свега име и опис. Име алата мора бити јединствено у листи алата и служи да се на њега реферише у упутствима агенту, док опис објашњава шта алат ради и даје контекст LLM-у за које задатке је тај алат користан [25]. Добро описани алати омогућавају моделу да из описа закључи када и како да их употреби. Поред тога, се могу дефинисати шеме аргумената за сложеније алате.

LangChain пружа више начина за регистровање алата. Најједноставнији је да имамо обичну Python функцију и да од ње направимо алат. Можемо користити класу Tool из LangChain библиотеке, којој прослеђујемо:

- **name:** назив алата (стринг),
- **func:** референцу на функцију коју извршава,
- **description:** опис намене алата.

Размотримо пример једноставног алата – функције која враћа тренутни датум. Прво ћемо дефинисати Python функцију, а затим је регистровати као алат са именом и описом:

```
from datetime import datetime
from langchain_core.tools import Tool

# Define function that returns current date in YYYY-MM-DD format
def get_current_date() -> str:
    """Returns current date as string in YYYY-MM-DD format."""
    return datetime.now().strftime("%Y-%m-%d")

# Create Tool object for tool registration
current_date_tool = Tool(
    name="get_current_date",
    func=get_current_date,
    description="Useful when agent needs today's date in YYYY-MM-DD
        format"
)
```

Структура 4.2: Регистрација простог алата за тренутни датум

Горњи пример кода 4.2 илуструје креирање прилагођеног алата. Функција *get_current_date* има једноставан документован опис који објашњава шта ради. Затим конструишемо Tool објекат коме дајемо јединствено име `get_current_date`, прослеђујемо саму Python функцију, и наведемо опис алата. Опис ће LangChain укључити у промпт агента тако да LLM зна да тај алат даје тренутни датум, те ће га позвати када кориснички задатак то захтева. После овакве регистрације, тај алат додајемо у листу алата које агент може користити (нпр. tools = [current_date_tool]). Уколико бисмо имали више алата, све их укључимо у ту листу коју касније предајемо агенту.

Коришћење меморије

Да би разговор са агентом био кохерентан и контекстуалан, неопходно је да агент има неку врсту меморије. Меморија омогућава да агент зна шта је већ речено у разговору, тако да може да користи тај контекст при генерисању следећих одговора. Без меморије, сваки упит би се третирао изоловано, што би довело до фрагментираних, неповезаних разговора [25]. LangChain зато обезбеђује класе за меморију које се могу укључити у ланце или агенте.

Постоји више врста меморије у LangChain-у, а две важне су:

- **ConversationBufferMemory** – меморија која чува комплетну историју разговора као низ порука (сваког питања корисника и одговора модела). Она представља најједноставнији облик: свака нова интеракција се додаје на крај меморијског буфера. Овај приступ осигурава да модел увек има комплетан контекст свих претходних размена, али може довести до великог броја токена ако је разговор дугачак.

- **ConversationSummaryMemory** – меморија која чува сажетак разговора уместо пуне историје. Користи LLM да након одређеног броја интеракција или када историја постане предуга, сажме досадашњи разговор у краћи резиме, који се затим користи као контекст уместо оригиналних порука. На тај начин се штеди на дужини контекста (мање потрошених токена), омогућавајући дуже разговоре [25]. Предност овог приступа је што агент може водити значајно дужи дијалог без губитка целокупног контекста, по цену ослањања на модел да направи добар сажетак.

Меморија је битна јер побољшава конзистентност агента: ако корисник постави пратеће питање, агент може да се позове на раније податке. На пример, ако корисник најпре пита „Какво је време у Токију данас?”, а затим „А шта је са сутрашњим временом?”, агент уз меморију разуме да се друго питање односи на временску прогнозу у Токију без да му се то експлицитно понови [25]. Без меморије, агент би сваки пут видео само појединачно питање и не би повезао да се “сутра” односи на исти град.

У пракси, додавање меморије агенту у LangChain-у је једноставно – при иницијализацији ланца или агента проследимо објекат меморије као параметар. Приликом сваког позива, LangChain ће аутоматски ажурирати меморију новим корисничким уносом и одговором модела. Укратко, интеграција меморије се изводи овако:

```
from langchain.memory import ConversationBufferMemory
from langchain.agents import initialize_agent, AgentType

# 1. Креирамо објекат меморије (овде бафер који чува целу историју)
memory = ConversationBufferMemory(memory_key="chat_history",
    return_messages=True)
```

```
6
7  # 2. Иницијализујемо LLM модел и листу алата (нпр. из претходне сек
       ције)
8  llm = ChatOpenAI(model_name="gpt-3.5-turbo", temperature=0)
9  tools = [current_date_tool]  # листа алата, претходно дефинисана
10
11 # 3. Иницијализујемо агента са меморијом
12 agent_chain = initialize_agent(
13     tools=tools,
14     llm=llm,
15     agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
16     memory=memory,
17     verbose=True
18 )
19
20 # 4. Сада агенту можемо поставити упит, а меморија ће се аутоматски
       ажурирати
21 response = agent_chain.run("Koji je danas datum?")
22 print(response)
```

Структура 4.3: Креирање агента са меморијом

У горњем примеру 4.3, користимо ConversationBufferMemory за чување историјата разговора (параметар memory_key="chat_history" одређује назив променљиве у шаблону промпта где ће се меморија убацити, a return_messages=True да меморија враћа листу порука уместо спојеног низа). Затим правимо агента помоћу функције initialize_agent, прослеђујући листу алата, LLM, тип агента и објекат меморије. Када позовемо agent_chain.run, агент ће добити приступ меморији и сваки његов одговор биће заснован на комплетном контексту разговора. После позива, memory објекат ће садржати и управо постављено питање и одговор, што омогућава агенту да на следеће питање одговори у контексту целе претходне конверзације.

Реактивно покретање агента

Реактивно покретање агента описан у секцији 2.4.3 је приступ за изградњу интелигентних агената који могу динамички да користе алате током резоновања. Уместо да агент само генерише одговор на основу унапред дефинисаног упита, ReAct агенти могу да размишљају корак по корак, доносе одлуке о томе које алате да позову и како да интерпретирају резултате тих позива у контексту задатка [26]. Овај приступ омогућава агенту да буде много флексибилнији и способнији за сложене задатке који захтевају више корака резоновања и интеракције са спољним изворима података.

У LangChain-у, да би агент радио у ReAct режиму, потребно је да буде конфигурисан

са списком алата које сме да користи, одговарајућим LLM моделом, и (опционо) меморијом за вођење дијалога. Такође му је потребан упит који имплементира ReAct логику. То обично специјално дизајниран шаблон који агенту даје инструкције да користи формат Мисао/Акција/Опажање. LangChain већ садржи уграђене упитне шаблоне за ReAct агенте, тако да их није нужно ручно писати. При креирању агента, може се искористити константу типа агента AgentType.ZERO_SHOT_REACT_DESCRIPTION, што означава управо ReAct агента који из описа алата бира који му је потребан (zero-shot приступ без претходних примера) [25]. У примерима, већ смо користили функцију initialize_agent са тим типом да направимо агента. Алтернативно, LangChain нуди и функцију create_react_agent за експлицитно прављење ReAct агента, као и класу AgentExecutor којом се финални агент покреће [25].

У пракси, initialize_agent(..., agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION, ...) враћа управо AgentExecutor спреман за коришћење.

Пример покретања агента у ReAct режиму је приказан у примеру кода 4.4. Под претпоставком да је агент са именом agent_chain који има један алат (get_current_date) и меморију већ иницијализован. Поставља се упит који захтева комбиновано резоновање и коришћење алата, на пример: “Који је данас датум и колико дана има до краја месеца?”. ReAct агент треба прво да схвати да му је потребан данашњи датум, да га дохвати, а онда да израчуна преостале дане у месецу. Покренимо агента:

```
query = "Који је данас датум и колико дана има до краја месеца?"
response = agent_chain.run(query)
print(response)
```

Структура 4.4: Покретање ReAct агента

Приликом оваквог позива, агент ће интерно генерисати низ “мисли” и “акција”. Могући ток би изгледао овако:

1. Мисао: “Питање тражи данашњи датум и број дана до краја месеца. Прво треба да сазнам данашњи датум.”
2. Акција: Позива алат get_current_date.
3. Опажање: Алат враћа, рецимо, “2025-09-19”.
4. Мисао: “Данас је 19. септембар 2025. Треба израчунати колико дана има до 30. септембра 2025.”.
5. Акција: (Није неопходна нова спољна акција, модел може сам да рачуна)
6. Опажање: (Није примењиво, рачун је унутрашњи)
7. Мисао: “Остало је 11 дана до краја септембра.”

8. Коначан одговор: “Данас је 19. септембар 2025. године, и преостало је 11 дана до краја месеца.”

Наравно, сам корисник добија само коначни одговор, док су кораци размишљања и акције извршене у позадини. Реактивно понашање овде се огледа у томе да агент није унапред знао које ће алате употребити или какву ће стратегију применити, већ је динамички одлучивао у ходу на основу самог питања и резултата међукорака. LangChain-ов ReAct агент уз помоћ AgentExecutor-а управља овим циклусом, што програмеру омогућава да једноставно проследи упит агенту као једну функцију, док се сложена логика одвија аутоматски. Резултат је снажан систем који може да обавља комплексне задатке комбинујући више корака резоновања и коришћења алата, дајући прецизне и контекстуалне одговоре [25].

4.3 Python EDGAR библиотеке за обраду 10-K извештаја

За потребе издвајања података, потребно је аутоматски приступати 10-K извештајима и из њих издвајати одређене информације које ће LLM анализирати.

Неколико истакнутих Python алата за рад са EDGAR подацима су:

- **edgar** библиотека: Једноставна Python библиотека која пружа интерфејс за приступ EDGAR архиви; омогућава претрагу по називу или CIK, преузимање најновијег 10-K и екстракцију текста [27].
- **EDGAR API клијенти:** Модерни клијенти за званични SEC API омогућавају претрагу поднесака и директно преузимање у JSON/XML облику.
- **Парсери SEC докумената:** *sec-parser* разлаже HTML 10-K на семантичке секције (наслови, пасуси, табеле), што поуздано омогућава извлачење релевантних делова попут “Item 1A. Risk Factors” [28].
- **SEC Extractor API (sec-api):** Комерцијални API и Python пакет за једноставно извлачење конкретних одељака (нпр. “Item 7”) и конверзију табела у структуре погодне за анализу [29].

У примени нашег агента, планирамо да користимо комбинацију наведених техника: аутоматизовано преузимање 10-K извештаја (edgar/SEC API), почетно парсирање (*sec-parser*/sec-api) и усмерено прослеђивање релевантних сегмената LLM-у. Овај корак претходне обраде смањује количину текста коју LLM анализира (трошак/прозор контекста) и повећава тачност података које модел разматра. Уз LangChain-ове документ лоудере и векторске базе, 10-K делови могу се индексирати ембединзима и претраживати семантички у RAG парадигми [23, 24].

На основу доступне литературе и алата, можемо илустровати могућности овакве обраде на конкретном примеру: узимање последњег 10-K извештаја компаније IBM и екстракција два различита дела. Прво, користећи *edgar* библиотеку, преузимамо HTML садржај извештаја и потом га прослеђујемо *sec-parser* алату који га дели на семантичке секције. Потом:

- Ако желимо квалитативне информације, попут описа пословања и ризика, можемо из парсираног стабла издвојити чланке означене као “Item 1. Business” или “Item 1A. Risk Factors” и добити чист текст тих секција. Ове секције често садрже више пасуса текста и могу бити обрађене LLM-ом ради сумирања или Q&A (нпр. „Како компанија описује утицај конкуренције на своје пословање у последњем 10-K?").

- Ако су нам потребне квантитативне информације, попут финансијских показатеља, можемо фокусирано извући табеларне делове 10-K. Коришћењем *sec-api* или мануелно уз *BeautifulSoup*, могуће је наћи табеле (нпр. биланс успеха, биланс стања) и конвертовати их у структуру погодну за анализу (рецимо DataFrame у Pandas-у). Тај бројчани контекст можемо проследити LLM моделу уз упутство да га протумачи (нпр. „На основу приложене табеле прихода и расхода из последње три године, објасни тренд прихода компаније.").

Закључно, Python EDGAR алати значајно проширују могућности LLM агента, омогућавајући му да буде специјализован за рад са извештајима. Комбинујући те податке са моћним разумевањем језика модела GPT-4, систем може да даје квалитетне одговоре – како описне (сумирање наративних делова), тако и аналитичке (поређење метрика) – на питања у вези са пословањем компанија. Овај интегрисани приступ спаја снагу LLM-а са прецизношћу доменских алата, што је кључно за научно-истраживачки рад који укључује обраду великих текстуалних финансијских података.

Агенти

Ово поглавље прелази са теоријских и инфраструктурних основа на конкретну архитектуру агената развијених у раду. Најпре се дефинише FinAsk агент: његова улога, општа архитектура и графички преглед. Након тога уводи се агент–судија као механизам евалуације квалитета одговора.

5.1 FinAsk агент

У овом одељку описан је дизајн LLM финансијског агента, уз образложење кључних одлука приликом његове изградње. Агент је осмишљен тако да аутономно спроводи анализу финансијских извештаја коришћењем више специјализованих алата. У наставку су наведене главне функционалности интегрисане у агента, као и разлози за њихово укључивање у систем.

5.1.1 Кључне функционалности агента

Ради испуњавања захтева домена, идентификовано је шест кључних алата које наш агент користи:

1. **Извлачење биланса стања:** Овај алат омогућава агенту да из финансијског извештаја издвоји структуриран биланс стања компаније. Парсирањем извештаја, агент проналази и извлачи главне категорије актива, пасива и капитала, враћајући их у облику погодном за даље рачунање. На тај начин, бројчани подаци о имовини, обавезама и капиталу постају директно доступни за анализу или одговарање на упите. Раздвајање биланса стања као посебног алата оправдано је тиме што гарантује прецизност у очитавању основних финансијских позиција и омогућава да се оне лако прослеђују другим компонентама система. За улаз прима период извештаја као и идентификатор паније, а излаз је приказан у наставку (Листинг 5.1).

2. **Извлачење биланса успеха:** Овај алат је задужен за прикупљање кључних података из биланса успеха. Агент путем њега дохвата приходе, расходе, нето добит,

Слика 5.1: Графички приказ архитектуре FinAsk агента

```
{
    "assets": {
        "current": 1200000,
        "non_current": 3000000,
        "total": 4200000
    },
    "liabilities": {
        "current": 1000000,
        "non_current": 2000000,
        "total": 3000000
    },
    "equity": {
        "total_equity": 1200000
    }
}
```

Структура 5.1: Структурирани биланс стања (пример излаза алата)

као и зараду по акцији (EPS) компаније. Ова функционалност је издвојена посебно јер обезбеђује да се показатељи пословног учинка прецизно извуку из извештаја. Структурирани приступ извлачењу прихода и расхода омогућава касније рачунање маржи, трендова и других метрика без ослањања на неструктурирани текст. За улаз прима период извештаја као и идентификатор паније, а излаз је приказан у наставку (Листинг 5.2).

```
{
    "revenue": 5000000,
    "expenses": 4200000,
    "net_income": 800000,
    "EPS": 2.5
}
```

Структура 5.2: Структурирани биланс успеха (пример излаза алата)

3. **Извештај о токовима готовине:** Трећи алат омогућава агенту да извуче податке из извештаја о токовима готовине. Конкретно, фокус је на три главне категорије новчаних токова: оперативни токови, инвестициони токови и финансијски токови. Агент издваја износ нето прилива/одлива готовине из пословања, улагања и финансирања, као и укупан нето промену готовине, што даје увид у то како компанија генерише и троши готовину. Укључивање овог алата је важно јер новчани токови пружају слику ликвидности и здравља пословања коју сами рачуни добити и губитка не откривају. Структурирани подаци о новчаним токовима омогућавају да се касније израчунају показатељи попут слободног новчаног тока или процени одрживост пословања. За улаз прима период извештаја као и идентификатор паније, а излаз је приказан у наставку (Листинг 5.3).

```
{
    "cash_flow_from_operations": 900000,
    "cash_flow_from_investing": -200000,
    "cash_flow_from_financing": 100000,
    "net_change_in_cash": 800000
}
```

Структура 5.3: Извод из извештаја о токовима готовине

4. **Извештај кључних показатеља:** Поред директног извлачења података, агент поседује алат за израчунавање кључних финансијских показатеља (енг. Key Performance Index - KPI) на основу прикупљених бројчаних вредности. Ова компонента рачуна индикаторе као што су повраћај на активу (енг. Return on Assets - ROA), повраћај на капитал (енг. Return on Equity - ROE), однос дуга и капитала (енг. debt-to-equity) и коефицијент текуће ликвидности (енг. current ratio). Алат преузима вредности из биланса стања и успеха које су добијене претходним алатима, и затим алгоритамски израчунава ове односе. Разлог за издвајање ове функционалности је да се обезбеди доследан и тачан прорачун финансијских индикатора, уместо да се ослонимо на LLM да их имплицитно рачуна из текста. Тако агент може кориснику директно да пружи увиде о профитабилности, ефикасности и ликвидности

компаније. Ове показатељи су израчунати на основу позиција у билансима. За улаз прима идентификатор паније и период извештаја, а излаз је приказан у наставку (Листинг 5.4).

```
{
    "ROA": 0.19,
    "ROE": 0.67,
    "debt_to_equity": 2.5,
    "current_ratio": 1.2
}
```

Структура 5.4: Пример израчунатих KPI показатеља

5. **Семантичко претраживање (RAG алат):** Планирамо класичан RAG (енг. Retrieval-Augmented Generation) модул који омогућава агенту да претражује спољни извор знања на основу семантичке сличности упита. Овај алат служи као допуна постојећим функционалностима у случају да корисничко питање захтева информацију која није директно доступна кроз претходне алате, агент може да претражи релевантне документе и извуче одговарајући контекст. У основи, ова компонента користи векторску репрезентацију текста и семантичку претрагу да би идентификовала најрелевантније пасусе, чиме се проширује знање агента изван унапред дефинисаних поља.

```
{
    "query": "Шта је проузроковало пораст прихода?",
    "result": {
        "text": "Менаџмент наводи да је пораст прихода резултат
        веће потражње на тржишту и успешног лансирања новог
        производа.",
        "source": "Годишњи извештај 2023 (MD&A одељак)"
    }
}
```

Структура 5.5: RAG упит и пронађени контекст

Овако осмишљен скуп алата покрива и квантитативне и квалитативне аспекте анализе финансијског извештаја. Сваки алат је укључен са јасном сврхом: прва три обезбеђују да се кључни финансијски подаци прецизно и структурирано извуку, четврти пружа аутоматизовану аналитику кроз прорачун показатеља. Пети алат (семантичка претрага) додатно осигурава да агент може да дође до сваке информације која би могла

бити упитана, а није директно обухваћена осталим алатима. Ова модуларна архитектура чини агента способним да одговори на разноврсна питања о финансијама компаније и оправдава сваку дизајнерску одлуку: специјализација алата повећава тачност, а њихова комбинација обезбеђује ширину покривених информација.

5.1.2 Подршка за меморију

Да бисмо омогућили будуће проширење функционалности, у плану је и додавање механизма меморије за агента. Подршка за меморију значи да ће агент бити у стању да памти релевантне информације током сесије или кроз више корака решавања задатка. На пример, агент би могао привремено чувати већ израчунате вредности или изведене закључке из ранијих корака анализе, како их не би прорачунавао више пута и како би разумео контекст сложенијих упита. Ова могућност је битна за сценарије у којима корисник поставља вишеструка повезана питања или када решавање једног задатка захтева низ међусобно повезаних корака. Додавањем меморије, агент постаје способан да води контекстуално свестан дијалог и да гради на претходно прикупљеним сазнањима, уместо да сваки пут креће од нуле.

Важно је, међутим, нагласити да у оквиру тренутне евалуације на скупу података меморија није пресудна компонента. Скуп задатака на којима тестирамо агента састоји се углавном од независних упита где сваки захтев може бити решен изоловано коришћењем наведених алата. То значи да агент, за потребе евалуације, не мора да памти резултате из једног упита да би решио следећи. Стога је подршка за меморију више унапређење ради будуће флексибилности него услов за тренутну функционалност. Ипак, предвиђањем оваквог механизма у дизајну, обезбеђујемо да је систем спреман за сложеније задатке и дијалошке сценарије који могу уследити након почетне фазе.

5.1.3 Стратегија резоновања

Приликом дизајнирања агента одлучено је за реактивни приступ резоновању. Ова одлука донета је након што је сагледана природа задатака у скупу података и сложеност упита. Већина захтева које агент треба да обради релативно су фокусирани (нпр. тражи се одређени финансијски показатељ или израчун на основу једног извештаја), те агенту обично није потребан дугачак низ корака да би дошао до одговора. Реактивни начин рада показао се довољним и ефикасним за овакве ситуације јер LLM може у једном кораку закључити који му је алат потребан (нпр. "извуци биланс стања", затим "обрачунај ROE") и добити одмах резултат, без формалног планирања целокупне стратегије. Поред једноставности, реактивни приступ је олакшао и развој и тестирање система, јер је лакше пратити и разумети ток одлучивања агента када он корак-по-корак образлаже и спроводи акције.

Насупрот томе, планско-извршни приступ би додао сложеност без јасне добити за

домен. Морала би бити имплементирана додатна фаза планирања и морало би се веровати да ће LLM увек правилно исцртати план, што у нашим експериментима није довело до бољих резултата. Стога је закључено да је реактивни приступ примеренији: агент је у стању да брже и поузданије дође до решења финансијских упита ослањајући се на постепено резоновање и прилагођавање, што се показало успешним у евалуацији.

5.2 Агент судија

Оцењивање квалитета одговора на комплексна финансијска питања представља изазов, јер традиционалне метрике (попут процентуалне тачности или преклапања са референцом) често нису довољно осетљиве. За отворена питања постоји више валидних начина да се пружи исправан одговор, тако да једноставна поређења низова речи могу заказати [30]. Са друге стране, ослањање на људске стручњаке за сваки одговор је споро и несразмерно скупо. Због тога се појавио приступ LLM-судија где се користи велики језички модел обучен да по унапред задатим критеријумима оцењује одговоре слично људском експерту [31, 30]. У контексту финансијских анализа заснованих на 10-K извештајима компанија, LLM-судија делује као виртуелни финансијски судија, процењујући тачност и квалитет понуђених одговора у односу на проверене "златне" одговоре (енг. gold standard) стручњака.

LLM као финансијски судија

LLM-судија функционише тако што му се, уз оригинално питање, проследе и два одговора: одговор кандидата (који треба оценити) и референтни (златни) одговор. Модел затим, пратећи детаљна упутства, упоређује кандидатски одговор са референтним и процењује у којој мери се подударају, односно где кандидатски одговор одступа. Овај приступ се назива оцењивање у поређењу са златним стандардом и уобичајен је у задацима отвореног типа одговора где се тачан одговор зна унапред [30]. У нашем случају, референтни одговор служи као поуздан оријентир на основу којег модел проверава чињенице, бројеве и логичку исправност у кандидатском одговору. На тај начин, LLM-судија може да "разуме" шта се очекује од идеалног одговора и да процени колико се дати одговор томе приближава. Овакав референтно вођен начин оцењивања показује се изводљивим и ефикасним, будући да су истраживања показала да модели-оцењивачи који имају увид у златни одговор могу донети оцене блиске људским судовима [31].

Важно је истаћи да је LLM који се користи као судија у финансијском домену прилагођен да разуме специфичну терминологију и контекст. То значи да модел поседује довољно доменског знања о финансијама и 10-K извештајима, како би могао да ухвати нијансе у одговорима. Уколико се уз питање и одговоре додају и релевантни изводи из 10-K извештаја (контекстуалне информације), модел их такође користи приликом

процене. Он проверава да ли се кандидатски одговор ослања на тај контекст и да ли је садржајно усаглашен са званичним подацима из извештаја, што додатно повећава тачност оцењивања.

Критеријуми оцене одговора

Оцењивање LLM-судије је структуирано према више унапред дефинисаних критеријума. У понуђеном упутству (prompt-у) експлицитно су наведени следећи аспекти, сваки са описом шта се очекује и скалом од 0 до 10:

1. **Чињенична тачност (Factual Correctness)** – Процена да ли су све изнете чињенице, финансијски подаци и бројке у кандидатском одговору тачни. Модел пореди конкретне цифре, процене и тврдње са референтним одговором и/или оригиналним подацима из 10-K. На пример, ако референтни одговор наводи приход од 5 милијарди долара, а кандидатски тврди 4 милијарде, LLM-судија ће то означити као фактичку грешку. Такође верификује прорачуне и да ли су извучени закључци у складу са чињеницама [32].

2. **Потпуност (Completeness)** – Оцењује да ли одговор обухвата све битне аспекте постављеног питања. Ако је питање сложено и има више подтема, добар одговор треба да се дотакне сваке од њих. LLM-судија упоређује обухват кандидатског одговора са дометом референтног: да ли је кандидат пропустио неки важан део или је дао површан одговор у односу на експертски [30]. Виша оцена подразумева да је кандидатски одговор скоро једнако свеобухватан као и референтни, док се низак скор даје ако одговор изоставља значајне делове.

3. **Финансијско расуђивање (Financial Reasoning)** – Мери квалитет логике и аналитичког процеса у одговору. Модел анализира да ли кандидат правилно примењује финансијске принципе и методологије, да ли су објашњења уз бројке смислена и да ли закључци следе из изнетих података. Једноставно речено, оцена расуђивања одражава да ли одговор размишља као финансијски аналитичар. Упоређује се приступ из кандидатског одговора са референтним образложењем: нпр. ако референтни одговор образлаже кретање прихода анализирајући тржишне услове, а кандидатски одговор износи број без објашњења, то указује на слабије расуђивање. Добар финансијски судија модел идентификује такве разлике у дубини анализа.

4. **Јасноћа (Clarity)** – Оцењује квалитет презентације одговора: да ли је одговор добро организован, језички јасан и лак за разумевање. Финансијски подаци могу бити сложени, па је важно да одговор буде представљен прегледно, са логичним током мисли. LLM-судија анализира да ли кандидатски одговор следи структуру

(нпр. увод, главни део, закључак), да ли користи примерен тон и стручну терминологију без непотребног жаргона, те да ли би циљна публика (нпр. инвеститори или пословни аналитичари) лако разумела излагање. Ово је једини критеријум који више гледа форму него садржину, али је важан за укупни утисак одговора.

Сваки од ових критеријума модел оцењује по десетостепеној скали, где 0 означава најлошији могући учинак (потпуно нетачан или неразумљив одговор), а 10 репрезентује изванредан квалитет (потпуно исправан, комплетан и јасан одговор, раван експертском). Поред појединачних оцена, LLM-судија на крају изводи и укупну оцену, која представља заокружен суд о одговору у целини. Уједно даје и образложење за сваку оцену – конкретна објашњења и примере који указују на јаке и слабе стране одговора. На пример, у образложењу за чињеничну тачност модел може навести: "Кандидат је тачно навео приход компаније за 2022. ($5,0 млрд према 10-K), али је погрешно приказао раст (тврди +15% уместо +8% годишње)". Таква детаљност помаже да се идентификују кључне разлике у односу на референтни одговор и пружи корисна повратна информација кандидатима.

Важно је нагласити да су ови критеријуми и њихове дефиниције део системског упутства прослеђеног моделу. Истраживања су показала да пажљиво осмишљени рубрици (скупови критеријума са јасним описима) значајно побољшавају конзистентност и објективност оцењивања од стране LLM модела [31]. У нашем случају, модел добија прецизне смернице шта да тражи у одговору по сваком аспекту, што смањује произвољност у суду модела. Ово одражава праксу из најновијих оквира за аутоматско оцењивање, где се више атрибута (попут тачности, релевантности, кохерентности, образложења итд.) узима у обзир ради темељније оцене [33]. Такав мулти-димензионални приступ омогућава моделу да сагледа одговор из више углова, слично као што би то учинио људски судија-експерт.

Процес оцењивања и логика модела

Када се LLM-судији проследе питање, кандидатски и референтни одговор (уз опционе контекстуалне информације из 10-K извештаја), активира се унапред дефинисан процес резоновања модела. Мада је овај процес унутрашњи и невидљив директно, можемо га описати у корацима које би модел логички могао предузети:

1. **Разумевање задатка и контекста:** Модел најпре прочита упутство које га поставља у улогу финансијског аналитичара-судије. Он "зна" да му је задатак да објективно упореди два одговора и оцени их према датим критеријумима. Затим пажљиво чита само питање како би разумео шта се тражи, као и додатне информације (нпр. изводе из 10-K) ако су обезбеђене. Ово осигурава да има пуни контекст пре него што погледа одговоре.

2. **Читање и поређење одговора:** Модел чита референтни (експертски) одговор и кандидатски одговор. Током овога, он вероватно сегментира информације по тематским целинама или ставкама (нпр. одговор може имати делове о приходима, трошковима, стратегији раста итд.). Модел пореди одговоре део по део: за сваку кључну чињеницу или аргумент из референтног одговора проверава да ли је и како покривен у кандидатском. Такође бележи ако кандидат наводи нешто што стручни одговор није (то може бити додатна тачка или можда грешка). Ово двосмерно поређење омогућава детекцију и пропуста (ствари које недостају у кандидатском), и додатака (ствари које је кандидат можда измислио или погрешно извео).

3. **Евалуација по критеријумима:** За сваки критеријум, модел примењује конкретна питања на прочитане одговоре:

    - **Чињенична тачност:** "Да ли се све бројке и тврдње у кандидатском одговору поклапају са информацијама из референтног одговора или датог контекста? Ако не, који су конкретни несагласни подаци?" Модел упоређује вредности један-на-један (нпр. финансијски показатељи, процентуалне промене) и означава разлике.
    - **Потпуност:** "Да ли кандидат покрива све што и експерт? Постоје ли делови питања на које није одговорио?" Модел идентификује сегменте из референтног одговора који немају еквивалент у кандидатском.
    - **Финансијско расуђивање:** "Да ли кандидат образлаже на сличан начин као експерт? Да ли је логика закључака исправна и поткрепљена подацима?" Овде модел проверава конзистентност аргумената и да ли кандидат изводи исте или сличне закључке.
    - **Нумеричка прецизност:** "Да ли су све рачунске операције тачне?" Модел може поновити једноставне прорачуне (ако су у одговору наведени) и проверити да ли кандидат није погрешно сумирао бројке или погрешно претворио валуте/јединице.
    - **Јасноћа:** "Да ли је одговор лако пратити и разумети? Постоји ли структура?" Овде модел процењује стил писања: нпр. да ли кандидатски одговор почиње прегледом, да ли су мисли логично повезане, да ли су реченице јасне или превише сложене.

4. **Синтеза налаза и додела оцена:** На основу горе наведених анализа, LLM-судија сажима налазе за сваки критеријум. Он у текстуалном облику образлаже шта је пронашао: нпр. "Под фактичком тачношћу: две вредности се не слажу са референтним одговором (грешка у нето приходу и стопи раста); Под потпуношћу: кандидат није поменуо анализу конкурентског окружења коју референтни одговор

садржи; ...” итд. После тога, модел преводи ове квалитативне налазе у нумеричке оцене 0-10 за сваки аспект. Оцењивање се врши у односу на референтни одговор као идеал: ако је кандидатски одговор готово идентичан по тачности и обухватности, добија високе оцене; ако значајно одступа, оцене су ниже. Коначно, модел одређује и укупну оцену, која није обичан просек већ целовита процена уз тежински осврт на све критеријуме.

5. **Структурирани излаз:** Резултат који LLM-судија враћа форматиран је као JSON објекат са пољима за сваки критеријум и објашњењима, унапред дефинисан као захтев у упутству. Ова структура осигурава конзистенцију и лакшу машинску обраду.

```
{
  "judge_evaluation": {
      "overall_score": 6,
      "criteria_scores": {
        "factual_correctness": 5,
        "completeness": 6,
        "financial_reasoning": 7,
        "clarity": 8
      },
      "criteria_explanations": {
        "factual_correctness": "The candidate lacks specific
        figures from the 10-K filing, making accuracy
        verification difficult.",
        "completeness": "The answer covers several aspects but
        lacks numerical data for complete analysis.",
        "financial_reasoning": "Candidate shows solid understanding
        of financial principles but lacks specific data.",
        "clarity": "Answer is well-structured and clearly
        communicates the analysis process."
      },
      "overall_explanation": "The answer provides general analysis
      but lacks specific numerical data and comparison with
      reference answer.",
      "confidence": 0.7,
      "judge_model": "gpt-4o",
      "judge_tokens": null
    }
}
```

Структура 5.6: Пример структурираног излаза LLM-судије

Валидација приступа и релевантна истраживања

Метод оцењивања одговора путем LLM-судије уз помоћ златних стандарда добио је велику пажњу у најновијим истраживањима. Резултати су охрабрујући: јаки LLM модели као што је GPT-4 показали су способност да се њихове оцене поклопе с људским експертским оценама у преко 80% случајева – што је отприлике на нивоу сагласности између два човека око истог одговора [31, 30]. Другим речима, аутоматски судија се у великој већини случајева слаже са тим да ли је одговор добар или лош исто као што би се сложили и независни људски процењивачи. Ово сугерише да је LLM-судија скалабилна и поуздана метода за приближно хумано оцењивање отворених одговора [30], нарочито када је модел подешен да ублажи одређене предрасуде.

Истраживања су такође указала на неке изазове и предуслове за успешност оваквог система. На пример, модели могу испољити позициони предрасуду (наклоност првом одговору ако се пореде два), предрасуду према опширности (склоност да више цене дуже, развучене одговоре) или самоповлашћивање (тенденцију модела да преферира сопствене генерисане одговоре) [31, 33]. У нашем оквиру, где се вреднује један кандидатски одговор у односу на референтни, брижљиво састављен prompt помаже да се те пристрасности умање – на пример, јасно се наводи да дужина одговора не сме утицати на оцену, како би се избегло неоправдано награђивање опширности. Такође, будући да је референтни одговор дат као оријентир, модел је вођен конкретним чињеницама, што смањује ризик од халуцинација или нагађања.

Детаљно дефинисање критеријума (попут нашег система са 5 категорија) показало се кључним за конзистентност: истраживање је уочило да у доменски специфичним задацима помаже дати моделу упутства шта чини добар одговор по сваком питању [32]. Наш систем претпоставља да су референтни (златни) одговори већ доступни за свако питање – што јесте додатни напор (припрема експертских одговора), али осигурава поуздано сидро за оцену. Алтернативно, за ситуације где нема комплетних златних одговора, могу се користити краће смернице (нпр. напомене за оцењивање) по питању [33], како сугерише најновија пракса, мада то излази из оквира овог рада.

Коначно, више студија је потврдило да моделски оцењивачи могу боље ухватити нијансе квалитета одговора него једноставне метрике тачности. Verga и сар. у свом раду истичу да оцене од стране LLM-а имају јачу корелацију са људским судом од рецимо буквалног поклапања са златним одговором [33]. Све ово пружа уверење да је наша поставка – LLM финансијски судија са рубриком и референтним одговором – утемељена на провереним методама и способна да пружи објективне и детаљне евалуације. То је нарочито корисно у домену финансија где је тачност информација критична, а истовремено одговори могу бити отвореног типа где крути критеријуми не дају пуну слику.

Потпуни шаблон упутства (prompt-а) који се користи за LLM-судију дат је у Додатку 7.

Резултати

У овом поглављу приказују се резултати евалуације финансијског агента на основу различитих критеријума квалитета. Анализа је спроведена за два приступа: основни модел и модел прилагођен за финансијски домен (FinAsk).

6.1 Поставка оквира за евалуацију

У оквиру експерименталне методологије развијена је Python скрипта који је коришћена за учитавање јавно доступног FinanceBench скупа података који садржи 150 финансијских питања и одговора. Процес евалуације је спроведен у неколико фаза: у првој фази су генерисани одговори од стране основног модела и FinAsk модела на постављена питања, при чему су сви одговори систематски забележени. У другој фази је активиран систем судије који је извршио евалуацију сваког појединачног одговора на основу референтних златних одговора из скупа података.

Критеријуми евалуације су обухватили четири кључна аспекта квалитета: јасноћу презентације, потпуност анализе, фактичку исправност и квалитет финансијског резоновања. Резултати евалуације су структурирано сачувани у JSON формату, што је омогућило даљу статистичку обраду и визуализацију. На основу прикупљених података генерисани су хистограми који приказују дистрибуцију скорова за сваки критеријум, омогућавајући компаративну анализу перформанси између различитих приступа.

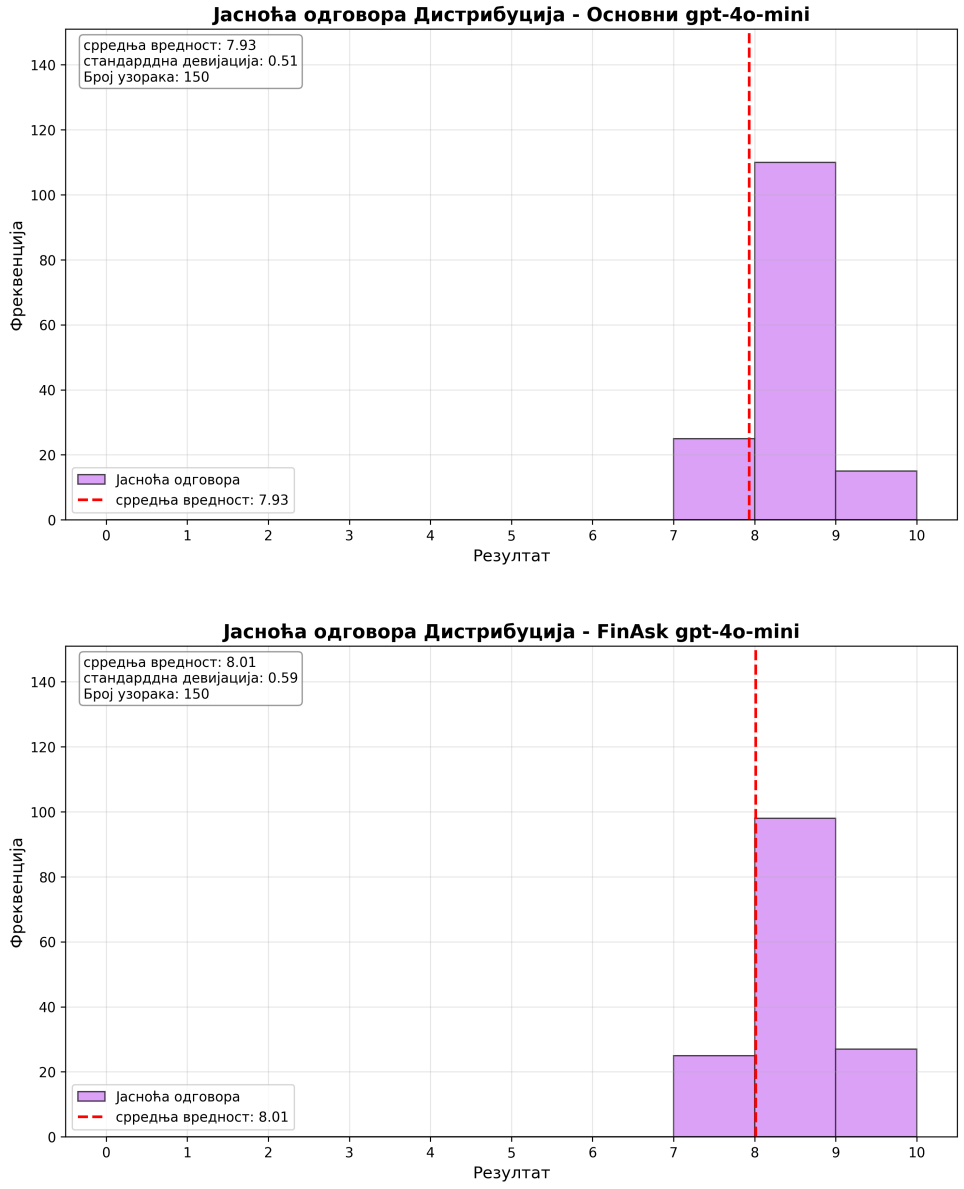
6.2 Анализа резултата

Следеће фигуре приказују поређење дистрибуције оцена за различите критеријуме евалуације између основног модела и FinAsk модела:

6.2.1 Анализа критеријума јасноће

Јасноћа одговора са средњим скором од 8.01 и најнижом стандардном девијацијом од 0.59 представља најконзистентнији аспект перформанси система. Ово указује на ефикасност у презентацији информација на начин који је разумљив крајњим корисницима,

што је есенцијално за практичну употребљивост финансијских агената. Међутим у односу на основни модел, не постоји значајна разлика у просечним оценама, што указује да оба модела имају сличан ниво способности у овом аспекту. Што говори да техникама прилагођавања промпта није постигнута значајна предност у односу на основни модел. Приказ поређења хистограма налази се на слици 6.1.

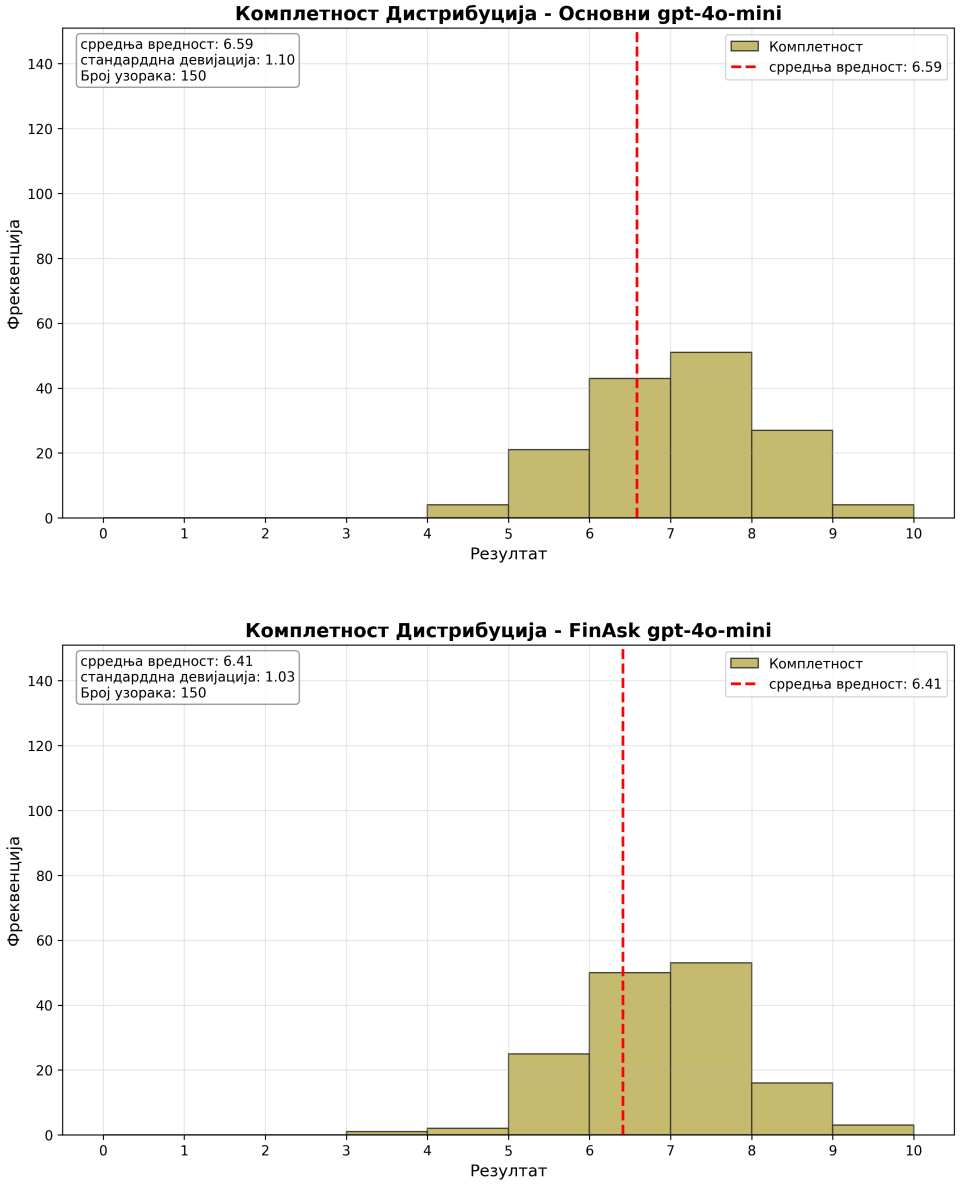


Слика 6.1: Хистограм анализе критеријума јасноће - горе: основни модел, доле: FinAsk модел

6.2.2 Анализа критеријума потпуности одговора

Комплетност одговора са средњим скором од 6.41 и стандардном девијацијом од 1.03 указује на релативно конзистентну перформансу у овом домену, што сугерише да архитектура система подржава систематичан приступ структурирању одговора. У поређењу

са основним моделом, FinAsk модел показује благо погоршање у просечним оценама, што указује на негативан утицај прилагођавања промпта на способност пружања обухватних анализа. Приказ поређења хистограма налази се на слици 6.2.

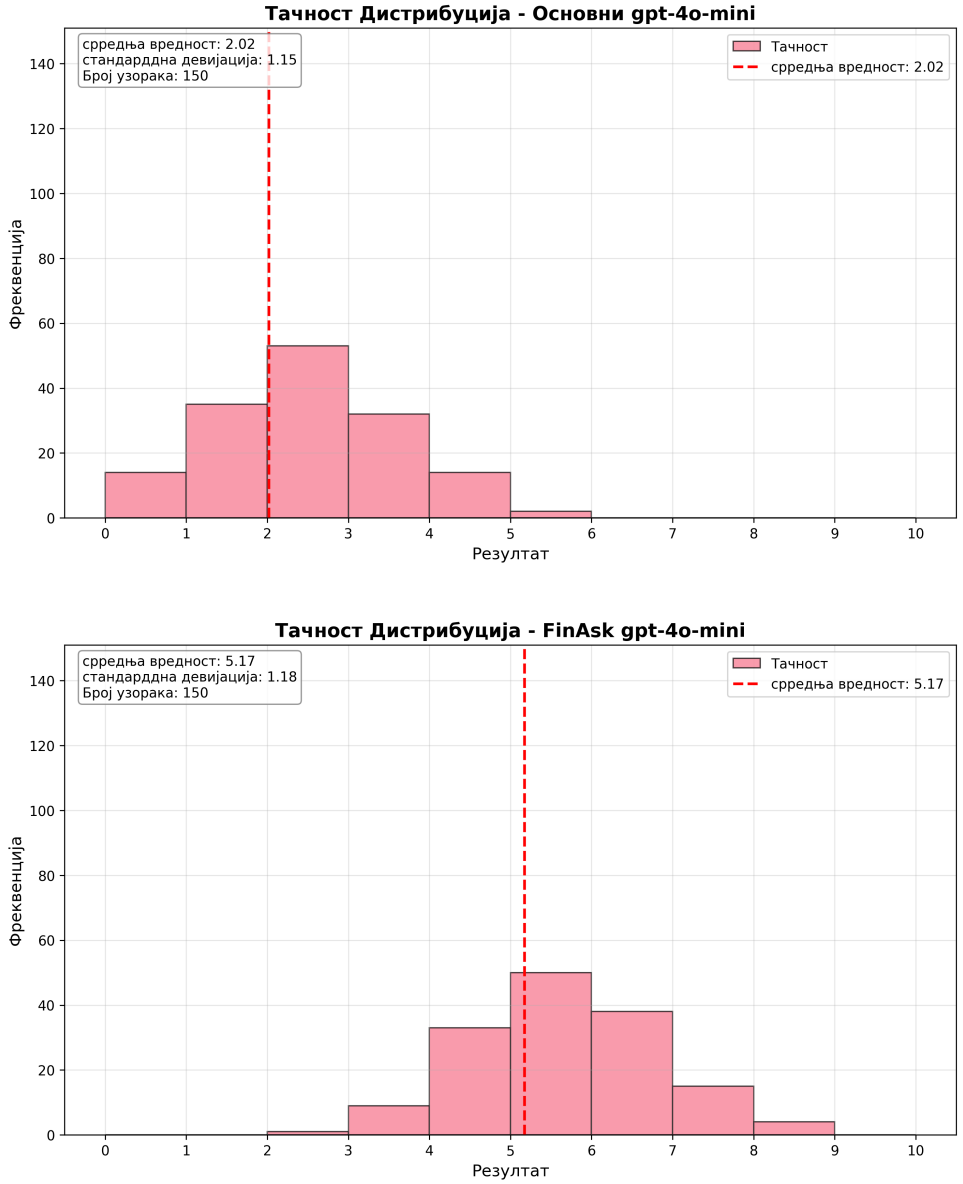


Слика 6.2: Хистограм анализе критеријума потпуности - горе: основни модел, доле: FinAsk модел

6.2.3 Анализа критеријума тачности

Фактуална исправност са просечним скором од 5.09 и стандардном девијацијом од 1.26 представља умерен ниво тачности који сугерише да систем успева да идентификује и репродукује релевантне финансијске информације у приближно половини случајева. Ова метрика је кључна за практичну примену система јер директно утиче на поверење корисника и употребљивост у реалним сценаријима финансијског саветовања.

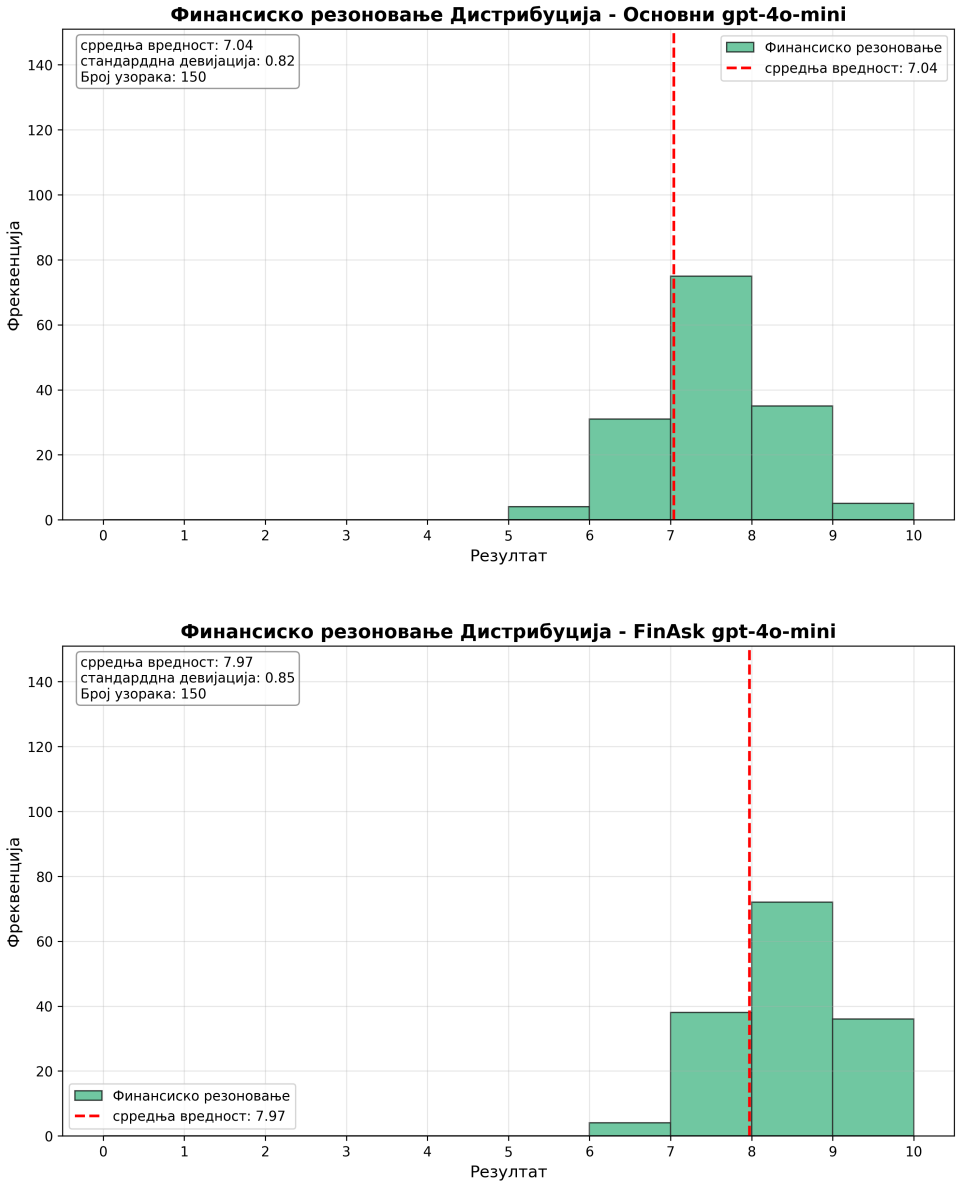
Релативно висока варијабилност указује на недоследност у приступу обради различитих типова упита, што може бити последица ограничења у основном језичком моделу или недовољне оптимизације промпт инжењеринга. У односу на основни модел приметно је велико побољшање што говори да основни модел нема информације које су потребне за тачан одговор, а FinAsk модел их успешно проналази. Додатно, ово показује способност агент асудије да распознаје и награди тачне одговоре. Приказ поређења хистограма налази се на слици 6.3.



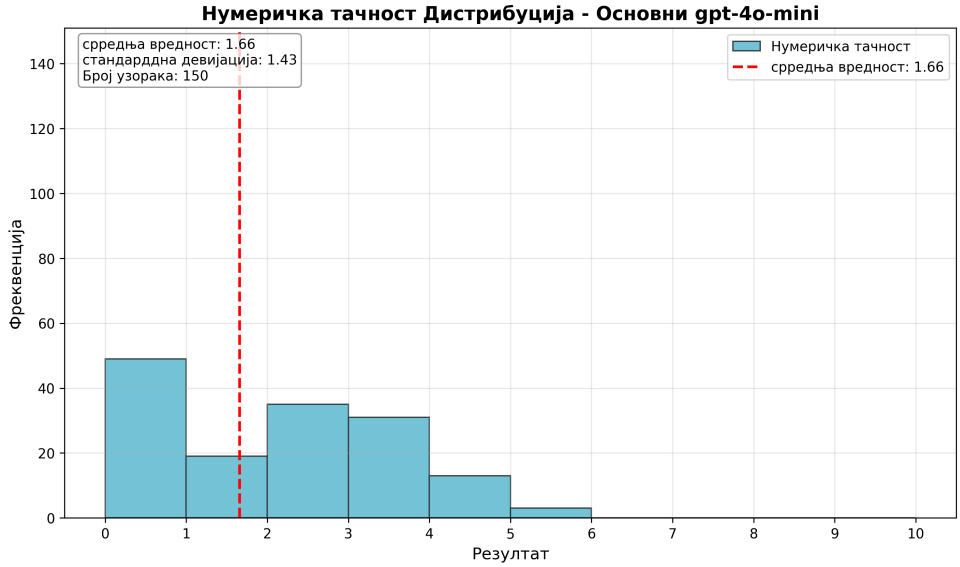
Слика 6.3: Хистограм анализе критеријума фактичке тачности - горе: основни модел, доле: FinAsk модел

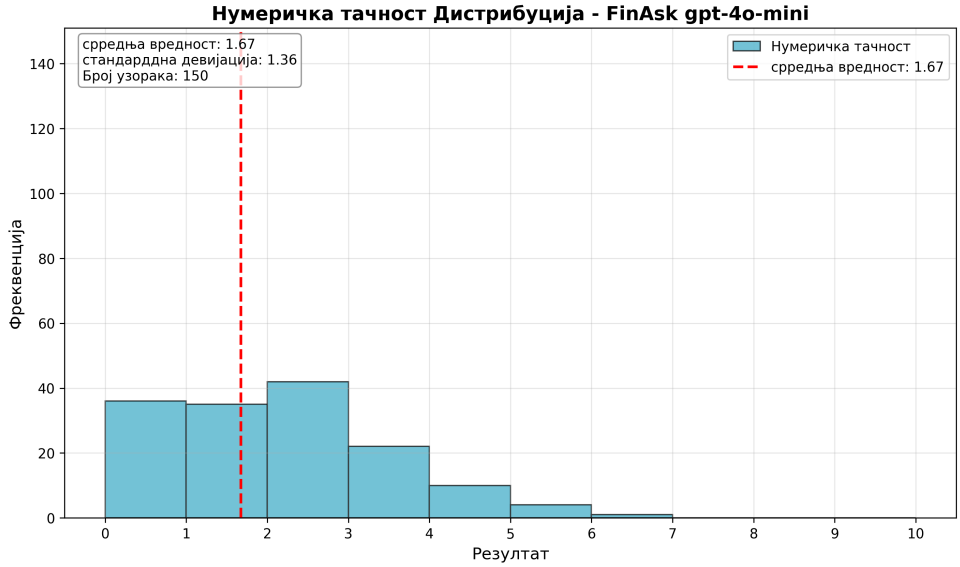
6.2.4 Анализа критеријума финансијског резоновања

К Финансијско резоновање са скором од 7.97 представља један од најснажнијих аспеката система, указујући на способност извођења логичких закључака и примене финансијских принципа у анализи. Ова висока перформанса је посебно значајна јер одражава дубину разумевања финансијских концепата и способност њихове примене у различитим контекстовима. Међутим, разлика у скору између основног модела и FinAsk модела је минимална, што указује да прилагођавање промпта није донело значајно побољшање у овој области. Поставља се питање да ли прилагођавање промпта може утицати на дубину финансијског резоновања или су ове способности уграђене у основни језички модел. Приказ поређења хистограма налази се на слици 6.4.



Слика 6.4: Хистограм анализе критеријума финансијског резоновања - горе: основни модел, доле: FinAsk модел





Слика 6.5: Хистограм анализе критеријума нумеричке тачности - горе: основни модел, доле: FinAsk модел

Закључак

У овом раду представљен је систем финансијског агента који користи велики језички модел (LLM) као судију за евалуацију својих одговора на финансијска питања. Главни циљ рада био је да се испита ефикасност овог приступа у побољшању квалитета одговора кроз прилагођавање промпта и ефикасност коришћења LLM-а као судије.

Резултати су показали значајно побољшање у тачности одговора када је коришћен прилагођени промпт у поређењу са основним моделом. Међутим није јасно да ли је агент показао могућност резоновања или је само боље проналазио информације у контексту. Поготово ако узмемо у обзир друге метрике где није било значајне разлике. Потребно је урадити дубљу анализу како би се утврдило да ли је побољшање последица бољег резоновања или само боље претраге информација. У будућности је потребно осмислити методологију која би подигла ниво тачности на виши ниво. Такође је потребна анализа колико су изабрани алати учинковити и да ли би избор других алата могао донети боље резултате.

Што се тиче коришћења LLM-а као судије, резултати основног модела су конзистентни са тачношћу у истраживању које је урађено на оргиналном FinanceBench скупу података [3], што указује на потенцијал овог приступа. За остале критеријуме је тешко одредити ефикасност овог приступа јер метрике могу бити субјективне. Потребно је урадити додатни преглед како би се боље разумело колико је LLM као судија поуздан у процени квалитета одговора. Генерално, упитно је колико се судија ослања на референтни одговор, а колико на своје знање и да ли је референтни одговор једини прави начин да се одговори на дато питање.

Списак скраћеница

- **API** – Интерфејс за програмирање апликација (енг. *Application Programming Interface*)
- **ВЈМ** – Велики језички модели (енг. *Large Language Models - LLM*)
- **BPE** – Кодирање парова бајтова (енг. *Byte Pair Encoding*)
- **EDGAR** – Програм електронског прикупљања, анализе и повлачења (енг. *Electronic Data Gathering, Analysis, and Retrieval*)
- **EPS** – Зарада по акцији (енг. *Earnings Per Share*)
- **GPT** – Генеративни претренирани трансформер (енг. *Generative Pre-trained Transformer*)
- **JSON** – JavaScript објект нотација (енг. *JavaScript Object Notation*)
- **KPI** – Кључни показатељи перформанси (енг. *Key Performance Indicators*)
- **LangChain** – Отворени оквир за оркестрацију LLM апликација
- **LLM** – Велики језички модели (енг. *Large Language Models*)
- **MSA** – Вишеглави механизам пажње (енг. *Multi-head Self-Attention*)
- **NLP** – Обрада природног језика (енг. *Natural Language Processing*)
- **PE** – Позиционо кодирање (енг. *Positional Encoding*)
- **ППНМ** – Потпуно повезане неуронске мреже (енг. *Fully Connected Neural Networks*)
- **Python** – Програмски језик високог нивоа
- **RAG** – Генерисање проширено преузимањем (енг. *Retrieval-Augmented Generation*)
- **ReAct** – Резоновање и делање (енг. *Reasoning and Acting*)
- **ROA** – Повраћај на активу (енг. *Return on Assets*)
- **ROE** – Повраћај на капитал (енг. *Return on Equity*)
- **SDK** – Комплет за развој софтвера (енг. *Software Development Kit*)

- **SEC** – Комисија за хартије од вредности САД (енг. *Securities and Exchange Commission*)
- **10-K** – Годишњи извештај америчких јавних компанија (енг. *Form 10-K*)
- **10-Q** – Квартални извештај америчких јавних компанија (енг. *Form 10-Q*)

Списак табела

Списак слика

Додатак

```
You are an expert financial analyst and judge evaluating answers to financial questions based
on company 10-K filings.

QUESTION:
{question}

CANDIDATE ANSWER (to evaluate):
{candidate_answer}

REFERENCE ANSWER (expert baseline):
{reference_answer}

{context_info}

EVALUATION CRITERIA:
1. **Factual Correctness (0-10)**: Are all financial facts, numbers, and claims accurate?
   - Compare specific figures, percentages, and metrics
   - Verify calculations and data points
   - Check for consistency with reference answer

2. **Completeness (0-10)**: Does the answer address all aspects of the question?
   - Consider all parts of the original question
   - Evaluate depth and breadth of coverage
   - Compare against reference answer scope

3. **Financial Reasoning (0-10)**: Is the reasoning process sound and financially literate?
   - Assess analytical approach and methodology
   - Evaluate logical flow and financial principles
   - Consider appropriateness of conclusions

4. **Numerical Accuracy (0-10)**: Are calculations and financial figures correct?
   - Verify mathematical computations
   - Check ratios, percentages, and metrics
   - Assess precision and units

5. **Clarity (0-10)**: Is the answer clear, well-structured, and easy to understand?
   - Evaluate communication effectiveness
   - Assess structure and organization
   - Consider professional presentation
```

```
EVALUATION GUIDELINES:
- Use the reference answer as the gold standard for comparison
- Consider the expert correctness and completeness rationale provided
- Be aware of the original methodology and system prompt used
- Focus on substance over style but maintain professional standards
- Allow for reasonable differences in expression while maintaining accuracy
- Provide specific, actionable feedback with examples

Provide your evaluation as a JSON object with the following structure:
{{
  "factual_correctness": {{
    "score": <0-10>,
    "explanation": "<detailed explanation with specific examples>"
  }},
  "completeness": {{
    "score": <0-10>,
    "explanation": "<detailed explanation comparing coverage>"
  }},
  "financial_reasoning": {{
    "score": <0-10>,
    "explanation": "<assessment of analytical approach>"
  }},
  "numerical_accuracy": {{
    "score": <0-10>,
    "explanation": "<verification of calculations and figures>"
  }},
  "clarity": {{
    "score": <0-10>,
    "explanation": "<assessment of communication effectiveness>"
  }},
  "overall_score": <0-10>,
  "overall_explanation": "<comprehensive assessment highlighting key strengths and weaknesses>",
  "confidence": <0-1>,
  "key_differences": ["<list of key differences from reference>"]
}}

Be thorough, fair, and provide actionable feedback.
```

Структура 7.1: Шаблон за рад агента судије

Литература

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

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

[7] 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.

[8] 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.

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

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

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

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

[18] P. Islam, A. Kannappan, D. Kiela, R. Qian, N. Scherrer, and B. Vidgen, "FinanceBench: A New Benchmark for Financial Question Answering," 2023. arXiv:2311.11944 [cs].

[19] OpenAI, "Function calling and other api updates." OpenAI News, 2023.

[20] D. Cheung, "Meta llama 2 vs. openai gpt-4: A comparative analysis." Medium, 2023.

[21] Anthropic, "Using anthropic: Best practices, parameters, and large context windows." PromptHub Blog, 2025.

[22] N. Kramer, "Deepseek: Everything you need to know about this new llm in one place." daily.dev, 2025.

[23] IBM, "What is langchain?." IBM Think Blog, 2023.

[24] A. Patriwala, "Langchain: A comprehensive framework for building llm applications." Medium, 2025.

[25] LangChain, "Langchain documentation." Online Documentation, 2024.

[26] S. Yao, D. Zhao, T. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[27] PyPI, "edgar library, v5.6.3." Python Package Index (PyPI), 2024.

[28] sec-parser documentation, "sec-parser: Semantic parsing for sec filings," 2023.

[29] sec-api.io, "Extract textual data from edgar 10-k filings using python." Tutorial, 2023.

[30] Evidently AI, "Llm-as-a-judge: a complete guide to using llms for evaluations." Evidently AI Blog, 2025.

[31] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging llm-as-a-judge with mt-bench and chatbot arena," *arXiv preprint arXiv:2306.05685*, 2023.

[32] Clearwater Analytics, "A cutting-edge framework for evaluating llm output." Clearwater Analytics Blog, 2023.

[33] P. Verga, N. Elaraby, S. Joshi, S. Min, D. Chen, and X. Ling, "Replacing judges with juries: Evaluating llm generations with a panel of diverse models," *arXiv preprint arXiv:2404.18796*, 2024.