**15 A. Vysvětlete pojem umělá inteligence. Vysvětlete aplikaci umělé inteligence ve formě LLM (large language model). Uveďte strukturu neuronové sítě používanou pro LLM a historii jejího vzniku. Uveďte zástupce SW umělé inteligence a oblast jejich použití (ChatGPT-jazykový model, Dall.e - tvorba obrazu, ….). Na příkladu ukažte využití AI v oblasti programování (proveďte slovní zadání a posuďte získaný kód).**

Umělá inteligence (AI) je oblast informatiky, která se zabývá vývojem systémů schopných vykonávat úkoly vyžadující lidskou inteligenci, jako je učení, rozhodování, rozpoznávání vzorů nebo zpracování jazyka. AI simuluje kognitivní procesy, jako je vnímání, logické myšlení nebo kreativita, a umožňuje strojům řešit složité problémy autonomně nebo s minimálním lidským zásahem.

Velké jazykové modely (LLM, Large Language Models) jsou typem AI založeným na hlubokém učení, které zpracovávají a generují lidský jazyk. LLM se trénují na obrovských textových datacích, aby porozuměly gramatice, sémantice a kontextu, což jim umožňuje:

* Generovat text (např. odpovědi na otázky, psaní esejí).
* Překládat jazyky.
* Odpovídat na otázky a poskytovat informace.
* Asistovat při programování (např. generování kódu).
* Vytvářet konverzační rozhraní (např. chatboty).

**Příklad použití**: ChatGPT od OpenAI je LLM, který dokáže vést dialog, odpovídat na otázky nebo pomáhat s úkoly, jako je psaní textů nebo analýza dat.

**Struktura neuronové sítě pro LLM a historie jejího vzniku**

**Struktura**

LLM jsou založeny na **transformerové architektuře**, což je typ hluboké neuronové sítě. Hlavní prvky transformeru zahrnují:

1. **Vstupní vrstva**: Text je převeden na číselné vektory (tokenizace a embedding), které reprezentují slova nebo části slov.
2. **Vrstvy kodéru a dekodéru**:
   * **Kodér**: Zpracovává vstupní text a vytváří jeho kontextovou reprezentaci (např. pro porozumění větám).
   * **Dekodér**: Generuje výstupní text na základě kontextu (např. při generování odpovědí).
3. **Self-attention mechanismus**: Klíčový prvek transformera, který umožňuje modelu zaměřit se na relevantní části vstupu při zpracování (např. porozumět vztahům mezi slovy v dlouhé větě).
4. **Feed-forward vrstvy**: Zpracovávají data v každé vrstvě transformera a přidávají nelinearitu.
5. **Výstupní vrstva**: Převádí zpracované vektory zpět na text (např. pravděpodobnosti slov).

Transformerové modely jsou škálovatelné, což umožňuje trénovat modely s miliardami parametrů (např. GPT-3 má 175 miliard parametrů).

**Historie vzniku**

* **1980–1990**: Počátky AI s jednoduchými neuronovými sítěmi a expertními systémy. Omezení kvůli nedostatečné výpočetní síle.
* **2013**: Word2Vec představil efektivní reprezentaci slov jako vektorů, což zlepšilo zpracování jazyka.
* **2017**: Google publikoval článek „Attention is All You Need“, který představil transformerovou architekturu. Ta nahradila starší modely (např. RNN, LSTM) díky efektivnímu zpracování dlouhých sekvencí.
* **2018–dosud**: Vývoj velkých modelů jako BERT (Google), GPT (OpenAI) a T5. Zvýšení velikosti modelů a trénovacích dat vedlo k výraznému zlepšení výkonu LLM.

**Zástupci SW umělé inteligence a oblasti použití**

1. **ChatGPT (OpenAI)**:
   * Typ: Jazykový model (LLM).
   * Použití: Konverzační rozhraní, asistence při psaní, odpovídání na otázky, vzdělávání.
2. **DALL·E (OpenAI)**:
   * Typ: Generativní model pro tvorbu obrazu.
   * Použití: Generování obrázků z textových popisů, design, umělecká tvorba.
3. **BERT (Google)**:
   * Typ: Model pro zpracování přirozeného jazyka.
   * Použití: Vyhledávání, analýza sentimentu, klasifikace textu.
4. **AlphaCode (DeepMind)**:
   * Typ: Model pro generování kódu.
   * Použití: Automatizace programování, řešení algoritmických úloh.
5. **Tesla AI**:
   * Typ: Model pro počítačové vidění a rozhodování.
   * Použití: Autonomní řízení vozidel, rozpoznávání překážek.

**Příklad využití AI v oblasti programování**

**Text k naučení**

Využití umělé inteligence v programování demonstruje její schopnost rychle generovat funkční kód na základě slovního zadání, což šetří čas a zjednodušuje vývoj. V příkladu jsme zadali AI úkol vytvořit program v Pythonu, který načte seznam čísel od uživatele, najde největší a nejmenší číslo, vypočítá průměr a ošetří možné chyby, jako je neplatný vstup. AI vygenerovala kód, který tyto požadavky splnil: použila vstupní funkci pro načtení čísel, metody max(), min() a sum() pro výpočty a blok try-except pro zachycení chyb, například zadání písmen místo čísel nebo prázdného seznamu. Tento příklad ukazuje, jak AI dokáže automatizovat rutinní programovací úkoly a přizpůsobit kód specifickým potřebám, což je užitečné zejména pro začátečníky nebo při rychlém prototypování.

Posouzení vygenerovaného kódu odhalilo jeho silné i slabší stránky, což pomáhá pochopit, jak AI funguje a jak ji efektivně využívat. Kód byl přehledný, funkční a robustní díky ošetření chyb, což zajistilo, že program nezhavaruje při neočekávaných vstupech. Přesto by mohl být vylepšen, například přidáním možnosti opakovat zadání při chybě nebo rozdělením logiky do funkcí pro větší modularitu. Tento proces ukazuje, že AI je výkonný nástroj, ale její výstup je třeba kontrolovat a případně upravit, aby vyhovoval pokročilejším požadavkům. Pro zapamatování je klíčové si uvědomit, že AI zjednodušuje psaní kódu, ale porozumění principům programování zůstává nezbytné pro ověření a optimalizaci výsledku.

**Zde rozepsaný příklad**

**Zadání**

Napište program v Pythonu, který:

* Načte seznam čísel od uživatele.
* Najde největší a nejmenší číslo v seznamu.
* Vypíše průměr všech čísel.
* Ošetří chyby (např. neplatný vstup).

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

**Posouzení kódu**

* **Funkčnost**: Kód splňuje všechny požadavky – načte čísla, najde maximum, minimum, vypočítá průměr a ošetřuje chyby (např. neplatný vstup, prázdný seznam).
* **Čitelnost**: Kód je přehledný, s jasnou strukturou a komentáři.
* **Ošetření chyb**: Použití try-except zajišťuje robustnost při zpracování neplatných vstupů (např. písmena místo čísel) nebo prázdného seznamu.
* **Možná vylepšení**:
  + Přidat validaci pro příliš dlouhý vstup.
  + Umožnit uživateli opakovat zadání při chybě (např. pomocí cyklu).
  + Oddělit logiku do funkcí pro lepší modularitu.