

Umělá inteligence

Implementace RAG pomocí DSPy a LangChain

Autoři: Natálie Rašková, Nela Bulavová, Filip Jína, Martin Šašinka

Obsah

[Úvod 2](#_Toc168505922)

[Cíl práce 2](#_Toc168505923)

[RAG 2](#_Toc168505924)

[DSPy 2](#_Toc168505925)

[LangChain 3](#_Toc168505926)

[Vytvoření chroma databáze 4](#_Toc168505927)

[Vyhledávání a generování 4](#_Toc168505928)

[Evaluace 5](#_Toc168505929)

[Budoucí práce 6](#_Toc168505930)

[Závěr 6](#_Toc168505931)

[Citace 7](#_Toc168505932)

[Seznam Obrázků 7](#_Toc168505933)

# Úvod

V této seminární práci se zaměříme na implementaci RAG modelu (Retrieval-Augmented Generation) pomocí GPT 3.5 TURBO. Budou použity dva frameworky: DSPy a LangChain. Práce zahrnuje jak teoretický popis, tak praktickou aplikaci a vyhodnocení výsledků. Studijní řád OU byl použit jako data pro RAG.

# Cíl práce

Cílem této seminární práce je vyzkoušet základní implementaci RAG modelu GPT 3.5 TURBO a pochopit, jak funguje zpracování dokumentů pomocí tohoto modelu. Zaměříme se na praktické aspekty nasazení modelu při práci s textovými dokumenty a na jeho schopnosti při vyhledávání a generování informací.

# RAG

RAG (Retrieval-Augmented Generation) je kombinací dvou hlavních přístupů v oblasti umělé inteligence: vyhledávání (retrieval) a generování (generation) textu. RAG je navržen tak, aby zlepšil kvalitu a přesnost odpovědí na dotazy tím, že nejprve vyhledá relevantní informace z externí databáze a poté na základě těchto informací vygeneruje odpověď.

**Jak funguje RAG model?**

**Vyhledávání:** Když je položen dotaz, model nejprve prohledá předem definovanou databázi dokumentů a vybere několik nejrelevantnějších úryvků textu.

**Generování:** Na základě vyhledaných úryvků textu model následně vygeneruje koherentní a informativní odpověď, která odpovídá položenému dotazu.

# DSPy

DSPy je rámec pro řešení pokročilých úloh s jazykovými modely (LM) a modely pro vyhledávání (RM). Sjednocuje techniky pro podněcování a dolaďování LM – a přístupy pro uvažování, sebezdokonalování a rozšiřování s vyhledáváním a nástroji (DSPy, *n.d.*).

Použití DSPy s RAG modelem

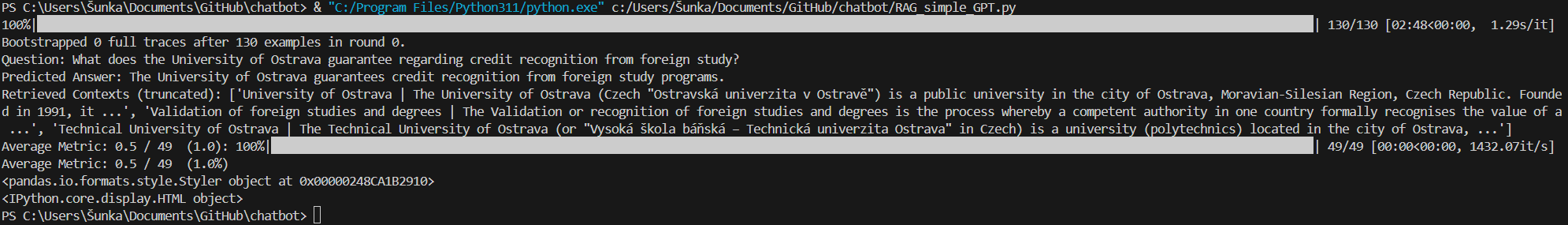
Naším původním záměrem bylo použít DSPy jako řešení pro implementaci RAG modelu Llama3. Chtěli jsme využít tento framework pro trénování a nasazení lokálního modelu pomocí aplikace Ollama s cílem zpracovávat vlastní dokumenty a zlepšit vyhledávání a generování odpovědí.

Bylo vytvořeno několik programů, tyto programy jsou přiloženy v příloze, které měli zaručit zařazení našich dat do dat modelu. Zde jsme bohužel vždy dospěli s modelem Llama3 ke stejnému výsledku. Parametr Bootstrapped 0 full traces na obrázku 1. má význam, že během daného kola bootstrapingu nebyly úspěšně vytvořeny a ověřeny žádné kompletní sledy kroků (tj. sekvence kroků od vstupu k výstupu). V kontextu modelu RAG to znamená, že model nedokázal generovat dostatečně přesné predikce, které by mohly být použity jako nové tréninkové příklady.Obsah obrázku snímek obrazovky, text, Písmo

Popis byl vytvořen automaticky

Obrázek 1: DSPy Llama3

Bohužel, tedy pravděpodobně z nedostatku zkušeností a technických znalostí, se nám nepodařilo dosáhnout uspokojivých výsledků. Trénování modelů neprobíhalo podle očekávání a výsledné modely nedosahovaly potřebné úrovně přesnosti a efektivity.

Obdobné výsledky jsme dosáhli po použití modelu GPT 3.5 TURBO, jak je možné vidět na obrázku 2.

Obrázek 2: DSPy GPT 3.5 TURBO

Tento neúspěch nás vedl k hledání alternativních řešení, což nás nakonec přivedlo k použití platformy LangChain.

# LangChain

LangChain je framework navržený pro práci s velkými jazykovými modely. Tento framework poskytuje nástroje pro snadnou integraci a správu modelů umělé inteligence, umožňuje vytváření databází a efektivní indexaci dat, což je klíčové pro aplikace jako je RAG.

Použití LangChain s RAG modelem

V našem projektu jsme použili LangChain pro implementaci RAG modelu GPT 3.5 TURBO s cílem zpracovat studijní řád Ostravské univerzity. LangChain nám umožnil:

**Vytvoření chroma databáze:** Uložili jsme a indexovali vektory z dokumentu studijního řádu, což umožnilo vyhledávání relevantních informací.

**Efektivní vyhledávání a generování:** LangChain využíval model GPT 3.5 TURBO k vyhledávání a generování odpovědí na základě dotazů, což nám pomohlo při analýze a interpretaci textu studijního řádu.

**Evaluace:** Provedli jsme hodnocení modelu pomocí metrik jako Score, Precision, Recall a F1-score, abychom posoudili přesnost a účinnost generovaných odpovědí.

Pro spuštění kódu je důležité si zobrazit a postupovat podle souboru README.md

Následující část textu bude popisovat tvorbu vektorové databáze dokumentu, vyhledávání a generování díky OpenAI GPT 3.5 TURBO a Evaluaci našeho RAG.

### Vytvoření chroma databáze

1. Příprava dokumentů

* Nejprve jsme extrahovali text z dokumentu studijního řádu.
* Tento text byl následně rozdělen na menší chunky (úseky zpracovatelné LM)
* Na obrázku 3 můžeme nastavit parametry chunk\_size (jak jednotlivé chunky budou velké) a chunk\_overlap (o kolik se jednotlivé chunky budou překrývat). Tyto parametry nám může výrazně měnit následné vyhledávání a generování v modelu.
* Obsah obrázku text, snímek obrazovky, Písmo

  Popis byl vytvořen automaticky

Obrázek 3: split\_text\_into\_chunks function

1. Generování vektorových reprezentací

* Každý chunk byl převeden na vektorovou reprezentaci (která nám umožní pozdější vyhledávání a indexaci informací)

1. Indexace vektorů

* Vygenerované vektory byly uloženy do chroma databáze. Chroma databáze je speciálně navržena pro ukládání a rychlé vyhledávání vektorových reprezentací.

## Vyhledávání a generování

1. Položení dotazu

* Uživatel položí dotaz týkající se informací ve studijním řádu Ostravské univerzity. Tento dotaz je vstupem pro celý proces vyhledávání a generování odpovědí.

1. Vyhledávání relevantních úseků

* LangChain nejprve prohledá chroma databázi, která obsahuje vektorové reprezentace úseků textu z dokumentu studijního řádu.
* Na základě podobnosti vektorů vyhledá nejrelevantnější úseky textu, které odpovídají položenému dotazu. Tím se zajistí, že model pracuje s nejvíce relevantními informacemi.

1. Generování odpovědí

* Model GPT 3.5 TURBO následně zpracuje vyhledané úseky textu a vygeneruje koherentní a informativní odpověď na základě dotazu.

Výsledky námi vyzkoušených promptů se nacházejí v příloze.

## Evaluace

Pro zhodnocení výkonu našeho modelu jsme provedli evaluaci pomocí několika klíčových metrik. Tyto metriky nám umožnily posoudit přesnost a účinnost generovaných odpovědí modelu GPT 3.5 TURBO při zpracování dotazů týkajících se studijního řádu Ostravské univerzity. Při evaluaci porovnáváme odpovědi vygenerované modelem s těmi předdefinovanými.

1. Skóre

* Skóre určuje, jak moc jsou vektorové reprezentace odpovědí podobné relevantním úsekům textu.
* Tato metrika využívá různé metody měření podobnosti, jako je kosinová podobnost, která porovnává úhel mezi dvěma vektory v prostoru.
* Vyšší skóre znamená vyšší podobnost, a tedy větší relevanci odpovědi.
* Skóre se může určit například Kosínovou podobností
* 

1. Precision (Přesnost)

* Precision měří, jaký podíl odpovědí vygenerovaných modelem je relevantní.
* 

1. Recall (Úplnost)

* Recall měří, jaký podíl všech relevantních odpovědí model správně identifikoval.
* 

1. F1-score

* F1-score je průměr hodnot precision a recall. Tato metrika poskytuje vyvážené měřítko výkonu modelu, zohledňující jak přesnost, tak úplnost.
* 

Výstup evaluací lze opět najít v příloze

# Budoucí práce

1. Optimalizace modelu

* Dále optimalizovat trénovací postupy a nastavení modelů pro dosažení lepších výsledků.
* Experimentovat s různými hyperparametry a technikami trénování pro zlepšení výkonu.

1. Zlepšení evaluace

* Použití pokročilejších metod evaluace a zahrnutí dalších metrik pro přesnější hodnocení výkonu modelů.
* Vytvoření většího souboru testovacích dotazů pro robustnější evaluaci.

1. Uživatelské rozhraní

* Navržení a implementace intuitivního uživatelského rozhraní pro interakci s modelem a databází.
* Poskytnutí nástrojů pro snadné zadávání dotazů a získávání odpovědí.

1. Vyzkoušení dalších frameworků pro RAG

* Experimentování s dalšími frameworky pro implementaci RAG modelů, jako jsou Haystack, Pyserini, nebo ElasticSearch.
* Srovnání jejich výkonu a efektivity s LangChain a DSPy.

1. Možnosti fine-tuningu s modelem

* Provedení fine-tuningu modelu GPT 3.5 TURBO na specifické datové sadě studijního řádu pro dosažení lepšího přizpůsobení a vyšší přesnosti odpovědí.
* Vyzkoušení různých metod fine-tuningu a jejich vliv na výkon modelu při zpracování dotazů.

# Závěr

Tato seminární práce se zaměřila na implementaci a evaluaci RAG modelu GPT 3.5 TURBO pomocí frameworků DSPy a LangChain s cílem zpracovat a analyzovat dokumenty, konkrétně studijní řád Ostravské univerzity. Nejprve jsme se pokusili využít DSPy pro trénování a nasazení lokálních modelů, avšak nedosáhli jsme uspokojivých výsledků, pravděpodobně kvůli nedostatku zkušeností nebo nedostatku kapacity modelu pracovat s českým jazykem. Následně jsme použili LangChain, který umožnil vytvoření chroma databáze pro efektivní ukládání a indexování vektorů z dokumentů. Tento přístup zlepšil vyhledávání a generování odpovědí na dotazy. Evaluace pomocí metrik jako Score, Precision, Recall a F1-score ukázala, že LangChain je efektivní platforma pro práci s GPT 3.5 TURBO. Projekt poskytl cenné poznatky o praktické implementaci RAG modelů, identifikoval oblasti pro další optimalizaci a rozšíření, a podtrhl význam technických znalostí při práci s pokročilými AI frameworky.

# Citace

DSPy. *Qdrant.tech* [online]. [cit. 2024-06-05]. Dostupné z: <https://qdrant.tech/documentation/frameworks/dspy/>

# Seznam Obrázků

[Obrázek 1: DSPy Llama3 3](#_Toc168505890)

[Obrázek 2: DSPy GPT 3.5 TURBO 3](#_Toc168505891)

[Obrázek 3: split\_text\_into\_chunks function 4](#_Toc168505892)