Univerzitet u Zenici

Politehnički fakultet

Softversko inženjerstvo

NLP - Modeli dubokog učenja za generisanje pitanja iz teksta (PDF-a i tekstualnog unosa)

Seminarski rad iz Vještačke inteligencije

Profesor: v. prof. dr. Elmir Babović

Asistent: v. as. doc. dr. Muharem Redžibašić

Student:

Ajla Brdarević, II-120

Zenica, oktobar 2024.

Bibliografska kartica rada

NAUČNO PODRUČJE RADA: Tehničke nauke

NAUČNO POLJE RADA: Softversko inženjerstvo

NAUČNA GRANA RADA: Vještačka inteligencija

USTANOVA U KOJOJ JE IZRAĐEN RAD: Univerzitet u Zenici, Politehnički fakultet

MENTORI RADA: v. prof. dr. Elmir Babović i v. as. doc. dr. Muharem Redžibašić

Sadržaj

Uvod

Ovaj rad predstavlja projekat automatskog generisanja pitanja iz tekstualnog sadržaja, iz unesenog teksta i PDF dokumenta. U današnjem digitalnom svijetu, količina informacija dostupnih u elektronskom formatu neprekidno raste, što stvara potrebu za efikasnim metodama obrade i analize tih informacija. Razvoj tehnologija dubokog učenja igra ključnu ulogu u postizanju ovog cilja, omogućavajući nam da stvorimo sofisticirane aplikacije koje mogu interpretirati i analizirati tekst.

Automatsko generisanje pitanja iz tekstova ne samo da olakšava učenje, već otvara i nove mogućnosti u obrazovanju. Ove aplikacije mogu izdvojiti ključne informacije iz teksta i formulisati relevantna pitanja koja korisnici mogu koristiti za provjeru svog znanja, pripremu za ispite ili dublje razumijevanje teme. U ovom radu biće prikazan kod razvijen za generisanje pitanja iz unesenih tekstova i PDF dokumenata, uz detaljno objašnjenje funkcionalnosti i primjene.

Implementacija modela je dostupna na: GitHub - ajla-brdarevic/pdf_question_generator: Project - Artificial intelligence

NLP - Modeli dubokog učenja za generisanje pitanja iz teksta (PDF-a i tekstualnog unosa)

1. Teorijski pregled

1.1. Obrada prirodnog jezika - NLP (Natural Language Processing)

„*NLP enables computers and digital devices to recognize, understand and generate text and speech by combining computational linguistics—the rule-based modeling of human language—together with statistical modeling, machine learning and deep learning.*" (Holdsworth, 2024)

Prema Holdsworth-u, Natural Language Processing (u daljem tekstu NLP) omogućava računarima i drugim digitalnim uređajima da prepoznaju, razumiju i generiraju tekst i govor. To se postiže kombiniranjem pravila lingvističkog modeliranja jezika sa metodama statičkog modeliranja, te mašinskog i dubokog učenja. Ovaj rad je primjer korištenja NLP-a jer se radi sa tekstualnim podacima, te samim tim dolazi do potrebe za razumijevanjem unijetog teksta. Krajnji rezultat trebaju biti smislene upitne rečenice, gramatički i pravopisno tačne za svaki korišteni jezik. Pitanja trebaju biti logična i smislena u okviru konteksta datog teksta, te formulisana na način koji omogućava jasan i relevantan odgovor.

„*Tekstna se lingvistika temelji na strukturalističkoj i generativnoj koncepciji jezika. Najznačajniji je pojam koherentnosti koji označuje gramatičke, sintaktičko-semantičke odnose među rečenicama.*" (Enciklopedija.hr, 2024)

Kod NLP-a, i u ovom radu, tekstna lingvistika, koja se odnosi na pravila lingvističkog modeliranja, je jako bitan parametar. Bez uzimanja u obzir pravila jezika i modeliranja, pitanja generisana mašinskim učenjem ne bi bila smislena, te tako ni korisna. Kroz implementaciju se mogla vidjeti značajna uloga pravilnog korištenja jezičkih pravila, s obzirom da i dalje postoje poteškoće kada se koristi bosanski jezik u datom tekstu ili PDF dokumentu. Još uvijek je potrebno štimanje i treniranje modela koji se koristio, upravo kako bi model mogao naučiti jezička pravila i norme, te kako bi sistem mogao generisati pravilna pitanja.

„*Machine learning is a type of artificial intelligence that performs data analysis tasks without explicit instructions. Machine learning technology can process large*

> *quantities of historical data, identify patterns, and predict new relationships between previously unknown data.*" (Amazon Web Services, Inc., n.d.)

Dakle, mašinsko učenje je oblik vještačke inteligencije. Ima mogućnost da obavlja analizu podataka bez potrebe za direktnim uputama. Ova tehnologija može procesirati velike količine historijskih podataka, identificirati obrasce, te prediktovati nove veze između podataka koji ranije nisu bili povezani. U kontekstu ovog rada, segment mašinskog učenja se odnosi na automatizovano prepoznavanje teksta iz PDF file-a, ali i na mogućnost generisana pitanja na osnovu poznatih podataka do tog trenutka. Pored toga, kroz fine-tuning se uče nove veze i obrasci između vrsta riječi te gramatičkih pravila ali i semantičkih značenja. Čak i kada model nije treniran na određenoj tematici, treba biti u stanju prepoznati pravila jezika.

NLP je ključna tehnologija u zadacima kao što su prepoznavanje govora, prevođenje jezika, analiza sentimenta i, generisanje pitanja iz teksta, kako je već rečeno. Tako, u ovom radu, NLP služi kao osnova za prepoznavanje semantičkih i sintaktičkih struktura unesenog teksta. Koristi se za tokenizaciju, prepoznavanje rečenica i analiziranje zavisnosti između riječi, čime omogućava modelu da formira relevantna pitanja.

> „*Tokenization is the process of dividing a text into smaller units known as tokens. Tokens are typically words or sub-words in the context of natural language processing… The process involves splitting a string, or text into a list of tokens… Tokenization involves using a tokenizer to segment unstructured data and natural language text into distinct chunks of information, treating them as different elements. The tokens within a document can be used as vector, transforming an unstructured text document into a numerical data structure suitable for machine learning.*" (GeeksforGeeks, 2019)

Kako je rečeno, tokenizacija je postupak dijeljenja teksta na manje jedinice poznate kao tokeni. U okviru obrade prirodnog jezika, tokeni su najčešće riječi ili dijelovi riječi. Ovaj proces uključuje razdvajanje niza ili teksta na listu tokena. Tokenizacija koristi tokenizator za segmentiranje nestrukturiranih podataka i teksta prirodnog jezika u zasebne dijelove informacija, tretirajući ih kao posebne elemente. Tokeni unutar dokumenta mogu se pretvoriti u vektore, čime nestrukturirani tekst postaje numerički podatkovni oblik pogodan za primjenu u mašinskom učenju. Tako je i u ovom radu tokenizacija bila ključan proces, jer bez toga ne bi

bilo moguće da algoritam prepoznaje tekst. Uz to, tokenizacija je bila potrebna tokom fine-tuning procesa, jer algoritam na ovaj način prolazi kroz skup podataka datih za treniranje i podešavanje već postojećeg modela.

NLP se oslanja na kombinaciju lingvističkih pravila i metoda mašinskog učenja kako bi analizirao jezik. U ovom slučaju, koristi se spaCy, jedna od najpoznatijih NLP biblioteka, za inicijalnu obradu teksta. Korišteni procesi omogućavaju lakše generisanje pitanja jer je tekst koji se analizira pravilno strukturiran i spreman za daljnju obradu.

> „*It's designed specifically for production use and helps you build applications that process and "understand" large volumes of text. It can be used to build information extraction or natural language understanding systems.*" (spacy.io, n.d.)

SpaCy pomaže u izgradnji aplikacija koje mogu obraditi i „razumjeti" velike količine teksta. Može se koristiti za razvoj sistema za ekstrakciju informacija ili razumijevanje prirodnog jezika. Zajedno u kombinaciji sa tokenizacijom za rezultat daje u potpunosti analiziran tekst, te na taj način je dobar temelj za stvaranje i smislenih pitanja, što se kroz implementaciju koda moglo primijetiti. Korištenjem ove biblioteke, čak i prije podešavanja i dodatnog treniranja postojećeg modela u ovom radu, rezultati su uglavnom bili smisleni i korisni. Nakon dodatnog štimanja modela, spaCy biblioteka je i dalje imala ulogu obrade ulaznog teksta za što bolje razumijevanje teksta, ponovo, zajedno sa tokenizacijom.

1.2. Primjena dubokog učenja u obradi jezika

> „*Deep learning is a type of machine learning that uses artificial neural networks to learn from data. Artificial neural networks are inspired by the human brain, and they can be used to solve a wide variety of problems, including image recognition, natural language processing, and speech recognition.*" (Google Cloud, n.d.)

Duboko učenje je podgrupa mašinskog učenja koja se zasniva na umjetnim neuronskim mrežama. Ove mreže simuliraju način na koji ljudski mozak obrađuje informacije. Kada se primjenjuje u NLP-u, duboko učenje omogućava računarima da uče iz ogromnih količina podataka i između ostalog prepoznaju obrasce u jeziku. Kroz fine-tuning, kako se bude dodavalo više različitih primjera, očekuje se brže i bolje učenje koristeći tehnike dubokog

učenja. Svakako, model bi bilo lakše trenirati na specifičnoj tematici, jer će tako biti u mogućnosti zaista razumjeti smisao i kontekst, pored same sintakse.

Ključni alat u dubokom učenju za obradu jezika su transformers modeli. Ovi modeli omogućavaju računarima da razumiju jezik u kontekstu, uzimajući u obzir ne samo riječi već i odnose između riječi unutar rečenica i paragrafa.

> „*A transformer model is a neural network that learns the context of sequential data and generates new data out of it… They are specifically designed to comprehend context and meaning by analyzing the relationship between different elements, and they rely almost entirely on a mathematical technique called attention to do so.*" (Ferrer, 2024)

Model transformers je vrsta neuronske mreže koja uči kontekst sekvencijalni podataka i na osnovu toga generiše nove podatke. Ovi modeli su posebno osmišljeni da razumiju kontekst i značenje analizirajući odnose između različitih elemenata. Razumijevanje i korištenje tokenizacije je ključno za iskorištavanje punog potencijala modela transformers-a u NLP-u.

Za generisanje pitanja iz teksta, korišten je model baziran na mT5 arhitekturi, o čemu će kasnije biti više riječi, te koja se zasniva na transformers tehnologiji. Ovaj model koristi slojeve neuronskih mreža koji analiziraju ulazni tekst, dekodiraju ga i na osnovu toga formiraju pitanja. Upravo u tom procesu, transformers tehnologija igra ključnu ulogu.

1.3. Generisanje pitanja - pretrenirani modeli mT5

Generisanje pitanja je zadatak koji koristi duboko učenje i pretrenirani NLP model koji je dodatno uštiman, kako bi automatski kreirao relevantna pitanja na osnovu unesenog teksta. Korišten je pretrenirani model Narrativa/mT5-base-finetuned-tydiQA-question-generation (Huggingface.co, 2024). Ovaj model je finotuniran (fine-tuned) za generisanje pitanja na brojnim jezicima.

> „*The document describes mT5 (multilingual T5) as a multilingual variant of the original T5 model, which was designed to work with a unified text-to-text format for various NLP tasks. mT5 is pre-trained on a dataset called mC4, which includes*

natural text from 101 languages sourced from the Common Crawl web scrape." (Xue et al., 2021)

mT5, ili „multilingual T5", je višelingvalna verzija originalnog T5 modela koji je dizajniran za rad sa jedinstvenim formatom „tekst-u-tekst" za različite zadatke u obradi prirodnog jezika (NLP). Ovaj model je unaprijed treniran na skupu podataka mC4, koji sadrži prirodni tekst na 101 jeziku prikupljen iz web izvora Common Crawl. Cilj mT5-a je zadržati prednosti T5 modela dok se prilagođava višelingvalnim kontekstima, čime postiže vrhunske rezultate.

Pretrenirani modeli poput mT5 treniraju se na ogromnim skupovima podataka s ciljem učenja univerzalnih obrazaca u jeziku. Prednost korištenja pretrenirano-finetuniranih modela je u tome što oni već imaju duboko znanje o strukturi jezika, što omogućava bržu implementaciju i bolje rezultate bez potrebe za dugotrajnim treniranjem na specifičnim skupovima podataka.

1.4. Fine-tuning mT5 modela

„*Fine-tuning in machine learning is the process of adapting a pre-trained model for specific tasks or use cases. It has become a fundamental deep learning technique, particularly in the training process of foundation models used for generative AI.*" (Bergmann, 2024)

Fine-tuning je proces prilagođavanja već treniranog modela za specifičnije potrebe. Predstavlja jednu od osnovnih tehnika dubokog učenja, posebno u procesu treniranja modela. U ovom radu je fine-tuning korišten kako bi se već trenirani model mT5 dodatno prilagodio. Iako je korišten model koji je već prošao kroz fine-tuning, krajnji rezultat postavljenih pitanja nike bio zadovoljavajući u potpunosti, te je bilo potrebno dodatno prilagoditi model.

Sam proces treniranja i štimanja će biti detaljno objašnjen kroz tehničku dokumentaciju sa primjerima koda. Biblioteke koje su bile potrebne za kreiranje skripte fine-tuning su pandas, transformers i datasets. Moduli/klase već spomenute biblioteke transformers koji su potrebni za rad:

- AutoModelForSeq2SeqLM

  "*As part of the Transformers library Hugging Face provides Auto Classes which works well when accessing popular models.*" (Niklas Heidloff, 2023)

  "*When we also add the decoder to create an encoder-decoder model, this is referred to as a sequence-to-sequence model or seq2seq for short. The model maps a sequence of one kind of data to a sequence of another kind of data.*" (huggingface.co, n.d.)

Dio transformers biblioteke obezbjeđuje auto klase koje pomažu kod pristupa velikog broja modela. Jedna od takvih klasa je i AutoModelForSeq2SeqLM. Seq2Seq model ili sekvenca-u-sekvencu model je model dubokog učenja koji prima jednu sekvencu podataka kao ulaz (npr. rečenicu na jednom jeziku) i generiše novu sekvencu kao izlaz (npr. rečenicu na drugom jeziku, ili riječi). Sastoji se od dva dijela - Encoder, koji kodira ulaznu sekvencu u skriveno stanje s ključnim informacijama, i Decoder, koji koristi to skriveno stanje za generisanje nove izlazne sekvence.

- AutoTokenizer

  "*Tokenizers are used to convert raw text into numerical tokens that can be understood by machine learning models. AutoTokenizer simplifies the process of selecting the correct tokenizer by automatically identifying the tokenizer associated with a specific pre-trained model.* " (IBM, 2023)

AutoTokinezer je također jedna od auto klasa. Dizajniran je da automatski odabere i učita odgovarajući tokenizator za dati unaprijed trenirani model. Tokenizatori pretvaraju sirovi tekst u numeričke tokene koje modeli mogu razumjeti, a AutoTokenizer pojednostavljuje ovaj proces prepoznavanjem tačno određenog tokenizatora povezanog s modelom.

- Trainer

  "*The Trainer is a complete training and evaluation loop for PyTorch models implemented in the Transformers library.*" (Huggingface.co, 2024)

Trainer je također jedan od modula ili klasa transformers biblioteke. Predstavlja petlju za trening bez potrebe za ručnim pisanjem potrebne petlje za trening. To omogućava brži trening, jer trainer modul obavlja automatski petlju i dosta pozadinskih automatizovanih procesa, što će se detaljno vidjeti u objašnjenu implementacije.

- TrainingArguments

  "*TrainingArguments is the subset of the arguments we use in our example scripts which relate to the training loop itself.*" (Huggingface.co, 2014)

TrainingArguments su podskup argumenata koji se koriste uz trainer, tj. trening petlju koja se automatski obavlja, kako je prethodno i objašnjeno.

- Datasets

  "*Datasets is a library for easily accessing and sharing datasets for Audio, Computer Vision, and Natural Language Processing (NLP) tasks.*" (Huggingface.co, 2024)

Datasets biblioteka se u ovom radu, kao i inače, koristi za olakšan pristup, rad i dijeljenje skupova podataka, između ostalog i za NLP. U ovom radu je potrebno koristiti datasets biblioteku tokom fine-tuninga, kako bi se moglo pristupiti i kako bi se obradio set podataka za treniranje.

Pandas

"*pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool…*" (Pandas, 2019)

Biblioteka pandas se koristi za analizu podataka i kao alat za manipulaciju. U ovom radu, pandas je potrebna biblioteka kako bi se pristupilo .csv file-u, te kako bi se podaci mogli obraditi.

2. Tehnička Dokumentacija

2.1. Struktura projekta

U glavnoj datoteci projekta nalaze se sljedeće datoteke i file-ovi:

fine_tuned_model:

- **config.json:** Konfiguracijski file za model mT5, podešen za generisanje pitanja iz skupa podataka tydiQA. Sadrži specifikacije arhitekture modela, tip tokenizatora, veličinu vokabulara i važne parametre.
- **generation_config.json:** Konfiguracijski file, sadrži postavke za model, uključujući decoder_start_token_id, eos_token_id, pad_token_id i verziju biblioteke transformers. Ove vrijednosti pomažu u definisanju funkcionalnosti modela tokom obrade podataka.

- **model.safetensors:** Sadrži sačuvane težine i konfiguraciju modela, koristeći safetensors format za poboljšanu sigurnost i efikasnost.
- **special_tokens_map.json:** Konfiguracija specijalnih tokena za model (eos_token, pad_token, unk_token) sadrži atribute koji definišu ponašanje pri obradi teksta, poput lstrip, rstrip i single_word, radi pravilne tokenizacije i sekvenciranja.
- **tokenizer_config.json:** Tokenizator konfiguracija definiše specijalne tokene (pad, eos, unk) i atribute poput max_length, padding_side i truncation_strategy, koji omogućavaju efikasno upravljanje popunjavanjem i skraćivanjem sekvenci za potrebe modela.
- **tokenizer.json:** Konfiguracija koja uključuje specijalne tokene, strategije za popunjavanje i skraćivanje, i atribute za obradu teksta, omogućavajući modelu pretvaranje ulaznog teksta u tokene za treniranje i obradu.

Results\checkpoint-30:

- **config.json:** Također konfiguracija u JSON formatu za model mT5, podešen za generisanje pitanja iz skupa podataka tydiQA.
- **generation_config.json:** Konfiguracija koja sadrži postavke za model.
- **model.safetensers:** Sadrži sačuvane težine i konfiguraciju modela, koristeći safetensors format za poboljšanu sigurnost i efikasnost.
- **optimizer.pt:** Sadrži težine i stanje optimizatora za treniranje PyTorch modela, omogućava nastavak treninga od istog stanja bez ponovne inicijalizacije optimizatora.
- **rng_state.pth:** Sadrži stanje generatora slučajnih brojeva (RNG) korištenog tokom treninga, omogućavajući tačno ponavljanje eksperimenata i rezultate, jer se svi slučajni procesi mogu identično reproducirati.
- **scheduler.pt:** Sadrži stanje planera stope učenja (learning rate scheduler) koji upravlja dinamikom promjene stope učenja tokom treniranja. Ovo omogućava nastavak treniranja bez gubitka informacija o prethodnim prilagodbama stope učenja, čime se poboljšava učinkovitost prilagodbe modela.
- **trainer_state.json:** Sadrži metapodatke o treniranju modela, uključujući broj epoha, globalne korake, korake evaluacije i logiranja, te postavke za čuvanje modela. Također uključuje kontrolne parametre za uvjete zaustavljanja epoha i evaluacija, dajući pregled trenutnog stanja i performansi modela.
- **training_args.bin:** Sadrži binarne podatke o argumentima za treniranje modela, uključujući ključne hiperparametre kao što su veličina serije, broj epoha i stopa

učenja. Omogućava ponavljanje treninga s istim parametrima, što olakšava reprodukciju eksperimenta i analizu performansi modela.

src:

- **pdf_utils.py**: Uvoz i obrada PDF file-a.
- **question_generation.py**: Generisanje pitanja koristeći novi fine-tuned model.
- **text_processing.py:** Pruža osnovu za obradu teksta na koristeći spaCy.

static:

- **style.css**: Sadrži CSS stilske definicije koje se koriste za oblikovanje web stranice.

templates:

- **index.html**: Glavni HTML element za web stranicu koja koristi Flask.
- **results.html**: Ovaj HTML element prikazuje rezultate generisanih pitanja.

.gitignore: Specificira šta neće biti uključeno u kontrolu verzija (Git).

app.py: Osnovu za web aplikaciju.

fine-tune.py: Dodatno obrada, treniranje i šitmanje već postojećeg modela.

requirements.txt: Navodi sve Python pakete koji su potrebni za izvršavanje projekta.

training_data.csv: Sadrži podatke za treniranje.

2.2. Detaljna razrada funkcionalnosti

2.2.1. Odabir i učitavanje PDF, ili unos teksta - app.py

Korisnik pristupa web aplikaciji preko http://127.0.0.1:5000. Odatle može odabrati iz padajućeg menija učitavanje PDF-a ili teksta, te jedan od ponuđenih jezika - engleski ili bosanski. Padajući meni je funkcionalnost u index.html koji uključuje HTML i JavaScript elemente, pomoću kojih korisnik bira prijenos PDF dokumenta, ili unos teksta u text box-u, kao i željeni jezik koji bi se trebao podudarati sa prenesenim PDF file-om ili unesenim tekstom. Veličina PDF file-a je postavljena na 16GB (Kod 1) u file-u app.py, koji je glavna aplikacija za pokretanje projekta.

```
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024
```

Kod 1. Maksimalna veličina učitane datoteke

U app.py, dostupna je funkcionalnost koja provjerava vrstu ulaza (input_type) koju korisnik šalje putem HTTP zahtjeva, tačnije padajućim menijem. (Kod 2)

1. Ako je odabran PDF - input_type 'pdf':

Uzimaju se file-ovi koji su poslati u zahtjevu pod ključem 'pdf_file'. Zatim se taj file proslijedi funkciji extract_text_from_pdf, koja izvlači tekst iz PDF dokumenta.

2. Ako je odabran tekst - input_type 'text':

   Uzimaju se podaci iz forme (text box) i uklanja se višak praznih prostora sa početka i kraja. Ako je rezultat prazan (tj. korisnik nije unio ništa), vraća se poruka „No text provided".

```
if input_type == 'pdf':
        file = request.files['pdf_file']
        text = extract_text_from_pdf(file)
    elif input_type == 'text':
        text = request.form.get('text', '').strip()
        if not text:
            return "No text provided"
```

Kod 2. Odabir i učitavanje PDF-a ili tekstualnog unosa

U kodu se tekst dobija putem request.form.get('text', '').strip(), čime se uklanjaju suvišni prostori. Prije generisanja pitanja, tekst se tokenizuje koristeći spaCy model. Funkcija nlp(text) razdvaja tekst na tokene (riječi i interpunkcijske znakove), a rezultat je objekat doc, s informacijama o tokenima. Na kraju, formira se string processed_text koji sadrži sve tokene spojene razmakom.

Tokenizacija se implicitno dešava i kada se koristi model za generisanje pitanja (question_generator). Kada model prima processed_text, on ga takođe tokenizuje kao dio svoje interne obrade pre nego što generiše pitanja. (Kod 3)

```
doc = nlp(text)
processed_text = " ".join([token.text for token in doc])
```

Kod 3. Tokenizacija i obrada teksta

2.2.2. Uvoz PDF file-a - pdf_utils.py:

Na početku se uvozi biblioteka PyPDF2. (Kod 4)

```
import PyPDF2
```

Kod 4. Učitavanje biblioteke PyPDF2

Funkcija extract_text_from_pdf(pdf_file) prima putanju do PDF datoteke kao argument i inicijalizira prazan string text za pohranu ekstraktovanog teksta. PDF se otvara u binarnom načinu čitanja koristeći with naredbu i učitava u objekt klase PdfReader iz biblioteke PyPDF2, koji omogućava pristup informacijama o PDF-u, uključujući tekst. U petlji se prolazi kroz sve stranice, a za svaku se poziva metoda extract_text(), koja vraća tekst. Ekstraktovani tekst se dodaje u string text uz novi red za razdvajanje. Na kraju, funkcija vraća string text sa kompletnim ekstraktovanim sadržajem iz PDF-a.(Kod 5)

```
def extract_text_from_pdf(pdf_file):
    text = ' '
    with pdf_file.stream as file:
        reader = PyPDF2.PdfReader(file)
        for page in reader.pages:
            text += page.extract_text() + '\n'
    return text  # Vraća kompletan ekstraktovani tekst iz PDF datoteke
```

Kod 5. Otvaranje i ekstrakcija podataka iz PDF-a

2.2.3. Obrada teksta - text_processing.py

Učitava se spaCy biblioteka i jezični modeli, zatim se definira funkcija process_text koja obrađuje tekst pomoću odgovarajućeg modela. Funkcija izvodi osnovne korake obrade teksta kao što su tokenizacija, lematizacija i part-of-speech tagging. S obzirom da je u trenutnom stanju projekta obrada teksta te ispravka u jezicima rađena kroz fine-tuning, ova skripta trenutno nema funkciju kao takva. (Kod 6)

```
import spacy

nlp_en = spacy.load("en_core_web_sm")
nlp_hr = spacy.load("hr_core_news_sm")

def process_text(text, language='en'):
    if language == 'en':
        doc = nlp_en(text)
    else:
        doc = nlp_hr(text)

    return doc.text
```

Kod 6. Obrada teksta sa spaCy

2.2.4. **Fine-tuning - fine_tune.py**

Prvi korak prije početka samog dodatnog podešavanja modela je stvaranje klasičnog .csv file-a u Excelu ili Google Sheets-u. U slučaju ovog rada, file-a training_data.csv sadrži dvije

kolone. Prva kolona - text, sadrži izjavne rečenice, dok druga kolona - question, sadrži pitanja. Gledajući po redovima, pored svake izjavne rečenice u jednoj koloni, u drugoj koloni se nalazi kako bi izgledalo pitanje vezano za izjavnu rečenicu iz prve kolone. U parovima, primjeri su prvo na bosanskom sa prefiksom [BS], te u narednom redu ista rečenica i pitanje na engleskom sa prefiksom [EN]. (Slika 1)

	A	B
1	text	question
2	[BS] Sarajevo je glavni grad Bosne i Hercegovine.	Koji je glavni grad Bosne i Hercegovine?
3	[EN] Sarajevo is the capital city of Bosnia and Herzegovina.	What is the capital city of Bosnia and Herzegovina?
4	[BS] Nikola Tesla je rođen 1856. godine u selu Smiljan.	Koje godine je rođen Nikola Tesla?
5	[EN] Nikola Tesla was born in 1856 in the village of Smiljan.	In which year was Nikola Tesla born?
6	[BS] Bosna i Hercegovina ima tri službena jezika: bosanski, hrvatski i srpski.	Koliko službenih jezika ima Bosna i Hercegovina?
7	[EN] Bosnia and Herzegovina has three official languages: Bosnian, Croatian, and Serbian.	How many official languages does Bosnia and Herzegovina have?
8	[BS] Mostar je poznat po Starom mostu, jednoj od najpoznatijih građevina u zemlji.	Po čemu je Mostar najpoznatiji?
9	[EN] Mostar is famous for the Old Bridge, one of the most recognizable landmarks in the country	What is Mostar famous for?
10	[BS] Bosna i Hercegovina se sastoji od dva entiteta: Federacije Bosne i Hercegovine i Republike	Od koliko se entiteta sastoji Bosna i Hercegovina?
11	[EN] Bosnia and Herzegovina consists of two entities: the Federation of Bosnia and Herzegovina	How many entities does Bosnia and Herzegovina consist of?
12	[BS] Rijeka Drina dijeli Bosnu i Hercegovinu i Srbiju.	Koja rijeka dijeli Bosnu i Hercegovinu i Srbiju?
13	[EN] The Drina River separates Bosnia and Herzegovina and Serbia.	Which river separates Bosnia and Herzegovina and Serbia?
14	[BS] Planina Bjelašnica je popularno odredište za skijanje u Bosni i Hercegovini.	Koja planina je popularno skijalište u Bosni i Hercegovini?
15	[EN] Mount Bjelašnica is a popular skiing destination in Bosnia and Herzegovina.	Which mountain is a popular ski resort in Bosnia and Herzegovina?
16	[BS] Nacionalna valuta Bosne i Hercegovine je konvertibilna marka (BAM).	Koja je nacionalna valuta Bosne i Hercegovine?
17	[EN] The national currency of Bosnia and Herzegovina is the Convertible Mark (BAM).	What is the national currency of Bosnia and Herzegovina?
18	[BS] Bosna i Hercegovina graniči sa Hrvatskom, Srbijom i Crnom Gorom.	S kojim zemljama Bosna i Hercegovina graniči?
19	[EN] Bosnia and Herzegovina borders Croatia, Serbia, and Montenegro.	Which countries does Bosnia and Herzegovina border?
20	[BS] Najviši vrh Bosne i Hercegovine je Maglić.	Koji je najviši vrh u Bosni i Hercegovini?
21	[EN] The highest peak in Bosnia and Herzegovina is Maglić.	What is the highest peak in Bosnia and Herzegovina?
22	[BS] Kognitivna disonanca je psihološki fenomen koji se javlja kada pojedinac osjeća nelagodu z	Šta je kognitivna disonanca?

Slika 1. Dio training_data.csv file-a koji sadrži trening podatke

Nakon što je file training_data spremljen u projekat, moguće je koristiti data set kako bi se model dalje trenirao. Ovaj data set ima 80 primjera, to jeste 40 pitanja koja su u paru bosasnki-engleski. Naravno, veći broj raznovrsnih primjera, daje bolja pitanja. Različita tematika, dužina rečenica, semantika, raznovrsne vrste riječi i širok vokabular, će imati veliki efekat na preciznost i kvalitetu pitanja. File se, uz dodatne funkcije, poziva u fine_tune.py file. Za rad sa .csv, potrebno je koristiti biblioteku pandas.

U ovom dijelu uvoze se potrebne biblioteke koje će se koristiti za fine-tuning modela. transformers biblioteka se koristi za rad s pretreniranim modelima i tokenizatorima, dok se datasets koristi za rad s podacima. pandas se koristi za manipulaciju podacima u CSV formatu. (Kod 7)

```
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer, Trainer,
TrainingArguments
from datasets import Dataset
import pandas as pd
```

Kod 7. Uvoz biblioteka za fine-tuning

Ovdje se definiše funkcija fine_tune_model, koja prima putanju do CSV fajla sa podacima. Sve operacije fine-tuninga modela će biti unutar ove funkcije. Definira se naziv modela koji će se koristiti za fine-tuning, a zatim se učitavaju pretrenirani model i odgovarajući tokenizator. (Kod 8)

```
def fine_tune_model(data_path):
    model_name = "Narrativa/mT5-base-finetuned-tydiQA-question-generation"
     model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
     tokenizer = AutoTokenizer.from_pretrained(model_name)
```

Kod 8. Funkcija za fine-tuning i uviz modela i tokenizatora

Podaci se učitavaju iz CSV fajla u pandas DataFrame. Zatim se pripremaju podaci za trening u formatu koji model može razumjeti, kombinujući tekst i pitanja iz DataFrame-a. (Kod 9)

```
    df = pd.read_csv(data_path)

    train_data = [
        {"text": text, "question": question}
        for text, question in zip(df['text'], df['question'])      ]
```

Kod 9. Učitavanje .csv file-a i priprema podataka za trening

Nakon pripreme podataka, konvertuju se u Dataset objekat koji se koristi za trening. (Kod 10)

```
train_dataset = Dataset.from_list(train_data)
```

Kod 10. Objekat koji se koristi za treniranje

Definiše se funkcija tokenize_function koja prima primjere i tokenizuje ulaze i izlaze (pitanja). Dodaje padding i skraćuje sekvence na odgovarajuće dužine. (Kod 11)

```
    def tokenize_function(examples):
        inputs = tokenizer(examples["text"], padding="max_length", truncation=True,
max_length=512)

        outputs = tokenizer(examples["question"], padding="max_length", truncation=True,
max_length=64)

        return {
            "input_ids": inputs.input_ids,
            "attention_mask": inputs.attention_mask,
            "labels": outputs.input_ids,
        }
```

Kod 11. Tokenizacija ulaznih i izlaznih podataka

Primjenjuje se funkcija tokenizacije na pripremljeni dataset u batch modusu, što znači da će podaci biti obrađeni u grupama radi efikasnosti. (Kod 12)

```
tokenized_dataset = train_dataset.map(tokenize_function, batched=True)
```

Kod 12. Tokenizacija na skupu podataka u grupama

Ovdje se definišu argumenti za trening, uključujući folder za izlazne rezultate, broj epoha, veličinu serije, korake zagrijavanja, težinsko smanjenje i direktorij za logovanje. (Kod 13)

```
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=3,
    per_device_train_batch_size=8,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
)
```

Kod 13. Argumenti za trening

Kreira se Trainer objekat koji upravlja procesom treninga. U njemu se navode model, argumenti za trening i tokenizovani dataset. (Kod 14)

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset,
)
```

Kod 14. Objekat za upravljanje treningom

Pokreće se proces treninga modela pomoću Trainer objekta. (Kod 15)

```
trainer.train()
```

Kod 15. Pokretanje treninga

Poslije treninga, model i tokenizator se čuvaju na disku za kasniju upotrebu. (Kod 16)

```
model.save_pretrained("./fine_tuned_model")
tokenizer.save_pretrained("./fine_tuned_model")
```

Kod 16. Spremanje modela i tokenizatora na disk

Ovdje se provjerava da li je skripta pokrenuta direktno. Ako jeste, poziva se funkcija za fine-tuning sa putanjom do podataka. (Kod 17)

```
if __name__ == "__main__":
    fine_tune_model("training_data.csv")
```

Kod 17. Provjera da li je skripta pokrenuta direknto

Pokretanje fine_tune.py, se generišu dvije datoteke, kako se vidi u kodu. U obe datoteke se nalaze većinski elementi za tokenizaciju i obradu teksta. S obzirom da su ti podaci automatski generisani, i praktično dio ugrađenih funkcionalnosti u razne biblioteke, kroz ovaj rad su već navedeni file-ovi, te ukratko navedene osnovne funkcionalnosti ili za šta su potrebni u dijelu „Struktura projekta".

2.2.5. Generisanje pitanja - question_generation.py

Uvoze se AutoModelForSeq2SeqLM i AutoTokenizer iz transformers biblioteke, koji će omogućiti učitavanje pretreniranog modela i odgovarajućeg tokenizatora. Definira se putanja do lokalno sačuvanog fino podešenog modela (model_path), a zatim se pomoću from_pretrained metoda učitavaju model i tokenizator. (Kod 18)

```
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
model_path = "./fine_tuned_model"
model = AutoModelForSeq2SeqLM.from_pretrained(model_path)
tokenizer = AutoTokenizer.from_pretrained(model_path)
```

Kod 18. Potrebne biblioteke i učitavanje podešenog i istreniranog modela

Ova funkcija generate_questions prima ulazni tekst i generira pitanja pomoću fino podešenog modela. Tokenizuje se ulazni tekst kako bi se dobili odgovarajući input_ids i attention_mask koji će biti korišteni za generisanje pitanja. Tokenizacija koristi padding i trunaciju kako bi osigurala da ulaz ne prelazi maksimalnu dužinu od 512 tokena. (Kod 19)

```
def generate_questions(text):
    inputs = tokenizer(text, return_tensors="pt", max_length=512, truncation=True)
```

Kod 19. Tokenizacija ulaznih podataka

Model generira više mogućih pitanja koristeći beam search strategiju, koja poboljšava generisanje sekvenci. Koriste se parametri kao što su maksimalna dužina generiranih sekvenci (max_length), broj traženih sekvenci (num_return_sequences), ograničenje na ponavljanje

istih ngrama (no_repeat_ngram_size), i zaustavljanje generisanja kada model postigne stabilnost (early_stopping). (Kod 20)

```
outputs = model.generate(
    **inputs,
    max_length=64,
    num_beams=5,
    num_return_sequences=5,
    no_repeat_ngram_size=2,
    early_stopping=True
)
```

Kod 20. Generisanje pitanja korištenjem beam search-a

Svaki izlazni niz dekodira se natrag u tekstualni format koristeći tokenizer.decode, pri čemu se preskaču specijalni tokeni (skip_special_tokens=True). Funkcija vraća listu generisanih pitanja. (Kod 21)

```
questions = [tokenizer.decode(output, skip_special_tokens=True) for output in outputs]
return questions
```

Kod 21. Dekodiranje i povrat generiranih pitanja

2.2.6. Prikaz rezultata - results.html

Rezultati se prikazuju u skladu sa definicijom u results.html (Kod 22), gde se prikazuje lista generisanih pitanja. U ovoj listi se nalaze generisana pitanja gdje je implementirana petlja koja prolazi kroz svako od tih pitanja, prikazujući ih kao stavke u listi.

```
<ul>
    {% for question in questions %}
      <li>{{ question }}</li>
    {% endfor %}
  </ul>
```

Kod 22. Lista generisanih pitanja u results.html

Zaključak

Zaključno, ovaj rad pruža uvid u proces automatskog generisanja pitanja iz tekstualnog sadržaja, s posebnim naglaskom na uneseni tekst i PDF dokumente. Kroz implementaciju razvijenog koda, demonstrirano je kako se mogu koristiti tehnologije dubokog učenja i obrade prirodnog jezika za efikasno analiziranje i generisanje pitanja iz različitih izvora informacija.

Korištenjem biblioteka kao što su PyPDF2 za ekstrakciju teksta iz PDF dokumenata, spaCy za obradu prirodnog jezika, i transformers za generisanje pitanja, ostvarena je funkcionalna aplikacija koja omogućava korisnicima da brzo dobiju relevantna pitanja na osnovu unijetog sadržaja. Ovaj pristup ne samo da povećava brzinu i efikasnost učenja, već i podržava korisničko angažovanje kroz interaktivni proces samoprovjere.

Pored navedenog, urađen je fine-tuning postojećeg modela. Time je unaprijeđen način generisanja pitanja, te je primijenjena i opisana procedura za fine-tuning ovog i sličnih modela. Jasno se kroz implementaciju mogao primijetiti napredak u obliku generisanih pitanja, koja su sa svakim dodavanjem trening-podataka, uspješnije i smislenije generisana.

Razvijeni kod pruža temelj za daljnji razvoj i unapređenje automatizovanih sistema u obrazovnom sektoru. Buduća istraživanja i razvoj mogu se fokusirati na optimizaciju algoritama za generisanje pitanja, proširenje jezičkih modela i detaljnije istraživanje mT5 postojećeg modela, različite forme pitanja npr. pitanja za višestruki izbor, kao i integraciju sa različitim platformama za e-učenje. Time se otvaraju nove mogućnosti za korištenje tehnologije u obrazovanju, čime se dodatno poboljšava kvalitet i dostupnost obrazovnih resursa.

Ovaj rad naglašava važnost interdisciplinarnog pristupa, spajajući oblasti kao što su računarske nauke, obrazovanje i lingvistika. Na taj način, doprinosi razvoju inovativnih rešenja koja mogu značajno unaprijediti proces učenja i podučavanja u digitalnom okruženju.

Literatura

- Holdsworth, J. (2024). What Is Natural Language Processing? IBM [online]. Dostupno na: https://www.ibm.com/topics/natural-language-processing (pristupljeno: 18. oktobar 2024)
- Enciklopedija.hr. (2024). tekstna lingvistika - Hrvatska enciklopedija. [online] Dostupno na: https://enciklopedija.hr/clanak/tekstna-lingvistika (pristupljeno: 26. 10. 2024].
- Amazon Web Services, Inc. (n.d.). What is Machine Learning? - Enterprise Machine Learning Explained - AWS. [online] Dostupno na: https://aws.amazon.com/what-is/machine-learning/ (pristupljeno 26.10.2024.)
- GeeksforGeeks. (2019). NLP | How tokenizing text, sentence, words works - GeeksforGeeks. [online] Dostupno na: https://www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-words-works/ (pristupljeno 19.10.2024.)
- spacy.io. (n.d.). spaCy · Industrial-strength Natural Language Processing in Python. [online] Dostupno na: https://spacy.io (pristupljeno: 26.10.2024.)
- Google Cloud. (n.d.). What is Deep Learning? Applications & Examples. [online] Dostupno na: https://cloud.google.com/discover/what-is-deep-learning (pristupljeno: 26.10.2024.)
- Ferrer, J. (2024). How Transformers Work: A Detailed Exploration of Transformer Architecture. [online] Datacamp.com. Dostupno nat: https://www.datacamp.com/tutorial/how-transformers-work (pristupljeno: 26.10.2024.)
- Huggingface.co. (2024). Narrativa/mT5-base-finetuned-tydiQA-question-generation · Hugging Face. [online] Dostupno na: https://huggingface.co/Narrativa/mT5-base-finetuned-tydiQA-question-generation (pristupljeno: 15. oktobar 2024).
- Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A. and Raffel, C. (2021). mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. [online] pp.483–498. Dostupno na: https://aclanthology.org/2021.naacl-main.41.pdf (pristupljeno: 26.10.2024.)

- Bergmann, D. (2024). What is Fine-Tuning? | IBM. [online] www.ibm.com. Dostupno na: https://www.ibm.com/topics/fine-tuning (pristupljeno: 26.10.2024.)
- Niklas Heidloff (2023). Causal LLMs and Seq2Seq Architectures. [online] Niklas Heidloff. Dostupno na: https://heidloff.net/article/causal-llm-seq2seq/ (pristupljeno: 27. 10. 2024)
- huggingface.co. (n.d.). Seq2Seq architectures - Hugging Face Audio Course. [online] Dostupno na: https://huggingface.co/learn/audio-course/chapter3/seq2seq (pristupljeno: 27. 10. 2024)
- community.ibm.com (2023). How to work with Pretrained Models with Transformers. [online] Ibm.com. Dostupno na: https://community.ibm.com/community/user/watsonx/blogs/ruslan-idelfonso-magaa-vsevolodovna/2023/10/05/how-to-work-with-pretrained-models-with-transforme (pristupljeno: 27. 10. 2024)
- Huggingface.co. (2024). Trainer. [online] Dostupno na: https://huggingface.co/docs/transformers/main/trainer (pristupljeno: 27. 10. 2024)
- Huggingface.co. (2014). Trainer. [online] Dostupno na: https://huggingface.co/docs/transformers/v4.46.0/en/main_classes/trainer#transformers.TrainingArguments (pristupljeno: 27. 10. 2024)
- Huggingface.co. (2024). Datasets. [online] Dostupno na: https://huggingface.co/docs/datasets/index?WT.mc_id=academic-105485-koreyst (pristupljeno: 27. 10. 2024)
- Pandas (2019). Python Data Analysis Library — pandas: Python Data Analysis Library. [online] Pydata.org. Dostupno na: https://pandas.pydata.org (pristupljeno: 27. 10. 2024)