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: doc.dr. Elmir Babović

Asistent: v.as. Muharem Redžibašić

Student:

Ajla Brdarević

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: doc. dr. Elmir Babović i dr.sc. 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

„NLP omogućava računarima i digitalnim uređajima da prepoznaju, razumiju i generišu tekst i govor kombinovanjem računalne lingvistike—modeliranja ljudskog jezika zasnovanog na pravilima—uz statističko modeliranje, učenje mašine i duboko učenje" (IBM, 2024).

NLP je ključna tehnologija u zadacima kao što su prepoznavanje govora, prevođenje jezika, analiza sentimenta i, u kontekstu ovog rada, generisanje pitanja iz teksta. Tako, u ovom radu, NLP služi kao osnova za prepoznavanje semantičkih i sintaktičkih struktura unesenog teksta. Koristi se za tokenizaciju[1], prepoznavanje rečenica i analiziranje zavisnosti između riječi, čime omogućava modelu da formira relevantna pitanja.

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.

1.2. Primjena dubokog učenja u obradi jezika

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 prepoznaju obrasce u jeziku.

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.

[1] Tokenizacija je proces razbijanja teksta na manje, značajne jedinice koje se nazivaju tokeni. Ovi tokeni mogu se koristiti kao numerički predstavnici teksta, što je neophodno za mnoge zadatke obrade prirodnog jezika. Tokenizacija je temeljna operacija koja utječe na sve kasnije korake u analizi teksta. - Menzli, A. (2021). Tokenization in NLP: Types, Challenges, Examples, Tools. [online] neptune.ai. Dostupno na: https://neptune.ai/blog/tokenization-in-nlp (pristupljeno: 19.10.2024)

Za generisanje pitanja iz teksta, korišten je model baziran na mT5 arhitekturi, koja se zasniva na transformers tehnologiji. Model mT5 (multilingvalni T5) je posebno dizajniran za zadatke obrade jezika na više jezika. Ovaj model koristi slojeve neuronskih mreža koji analiziraju ulazni tekst, dekodiraju ga i na osnovu toga formiraju pitanja.

Primjena dubokog učenja u obradi jezika omogućava efikasno rješavanje složenih zadataka poput generisanja pitanja jer model može samostalno naučiti pravilne gramatičke strukture, odnose među riječima i kako pravilno postaviti pitanje iz danog konteksta.

1.3. Generisanje pitanja - pretrenirani modeli mT5

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

Model mT5 je pretrenirani model koji se oslanja na Google-ov mT5 i treniran je na mC4 korpusu, pokrivajući 101 jezik. Kako bi se izbjeglo dodatno treniranje modela za postavljanje pitanja, isprobano je nekoliko već pretreniranja modela za ovaj projekat, ali se Narrativa/mT5 pokazao kao najbolji, s obzirom na to da su format i prirodnost generisanih pitanja najoptimalniji.

Pretrenirani modeli poput mT5 treniraju se na ogromnim skupovima podataka s ciljem učenja univerzalnih obrazaca u jeziku. Ovi modeli koriste se u raznim NLP zadacima, kao što su prevođenje jezika, sumiranje teksta i, u ovom slučaju, generisanje pitanja. 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.

U generisanju pitanja, model uzima ulazni tekst i dekodira ga kako bi iz njega izvukao ključne informacije potrebne za formulisanje pitanja. Primjenom tehnike transfer learning, ovaj pretrenirani model je finotuniran za određeni zadatak—u ovom slučaju, generisanje pitanja—čime je optimizovan za specifične zadatke bez potrebe za treniranjem od nule.

Za višelingvalnu podršku, model je sposoban generisati pitanja na više jezika, što čini projekt fleksibilnim. Korisnik može odabrati jezik (bosanski ili engleski), a model će automatski generisati pitanja na odabranom jeziku.

2. Tehnička Dokumentacija

2.1. Tehnologije

- Flask: Web okvir za Python koji se koristi za izgradnju web aplikacija.
- Transformers (Hugging Face): Biblioteka koja omogućava korištenje pretreniranih modela za NLP, uključujući T5 model za generisanje pitanja.
- PyPDF2: Biblioteka za ekstrakciju teksta iz PDF dokumenata.
- spaCy: Biblioteka za obradu prirodnog jezika koja se koristi za tokenizaciju.

2.2. Struktura projekta

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

- **src:**
    - **pdf_utils.py**: Na početku se uvozi biblioteka PyPDF2. Ova biblioteka je specijalizirana za rad s PDF datotekama u Pythonu i omogućava čitanje i obradu njihovog sadržaja. Zatim se definira funkcija extract_text_from_pdf(pdf_file). Ova funkcija prima kao argument putanju do PDF datoteke. Unutar funkcije se inicijalizira prazan string text u koji će se pohranjivati ekstraktovani tekst. Koristeći with naredbu, PDF datoteka se otvara u binarnom načinu čitanja. Ovaj način omogućava čitanje binarnih podataka, kao što su PDF datoteke. Otvorena PDF datoteka se učitava u objekt klase PdfReader iz biblioteke PyPDF2. Ovaj objekt nam daje mogućnost pristupa različitim informacijama o PDF-u, uključujući i njegov tekstualni sadržaj. U petlji se iterira kroz sve stranice PDF dokumenta. Za svaku stranicu se poziva metoda extract_text() koja vraća tekst sa te stranice. Ekstraktovani tekst sa svake stranice se dodaje u string text, uz dodavanje novog reda na kraju kako bi se razdvojili tekstovi sa različitih stranica. Na kraju funkcije se vraća string text koji sadrži kompletan ekstraktovani tekst iz PDF datoteke.
    - **question_generation.py**: Ovaj Python kod koristi biblioteku transformers kako bi generirao pitanja iz danog teksta. To radi importovanjem pipeline-a -

transformers.pipeline koji je jednostavan način za korištenje prethodno obučenih modela za različite zadatke obrade prirodnog jezika. Učitavanje modela sa question_generator - varijabla koja sadrži model obučen specifično za generisanje pitanja iz teksta. Ovaj model je "Narrativa/mT5-base-finetuned-tydiQA-question-generation", što znači da je baziran na mT5 modelu i fino podešen na zadatku generisanja pitanja koristeći skup podataka TyDiQA. Funkcija generate_questions prima tekst kao ulaz i vraća listu generisanih pitanja. Generisanje pitanja - kada pozovemo funkciju, model će analizirati dati tekst i generisati nekoliko pitanja koja se odnose na taj tekst.

    - **text_processing.py:** Ovaj Python kod pruža osnovu za obradu teksta na koristeći spaCy. 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.

- **static:**
    - **style.css**: Ova datoteka sadrži CSS stilske definicije koje se koriste za oblikovanje web stranice, to jeste oba HTML dokumenta.
- **templates:**
    - **index.html**: Ovo je glavni HTML element za web stranicu koja koristi Flask. Sadrži sve potrebne HTML elemente za izgled i funkcionalnosti stranice - odabir i učitavanje PDF file-a ili teksta, te dugme za početak generisanja pitanja.
    - **results.html**: Ovaj HTML element prikazuje rezultate generisanih pitanja.
- **.gitignore**: Ova datoteka specificira koje datoteke i file-ovi neće biti uključeni u kontrolu verzija (Git).
- **README.md**: Ova datoteka sadrži opće informacije o projektu, upute za instalaciju i korištenje, kao i druge relevantne detalje.
- **app.py**: Ovaj Python kod predstavlja osnovu za web aplikaciju koja može generisati pitanja na temelju danog teksta ili PDF dokumenta. Koristi Flask okvir za izgradnju aplikacije, spaCy za obradu prirodnog jezika i druge biblioteke za ekstrakciju teksta, obradu teksta i generisanje pitanja. Aplikacija omogućava korisnicima da učitaju PDF datoteku ili unesu tekst, a zatim generiše pitanja na temelju tog unosa.

- **requirements.txt**: Ova datoteka navodi sve Python pakete koji su potrebni za izvršavanje projekta.

2.3. Detaljna razrada funkcionalnosti

- **Odabir i učitavanje PDF, ili unos teksta**

Korisnik pristupa web aplikaciji preko http://127.0.0.1:5000. Odatle korisnik može odabrati iz padajućeg menija da li će učitati PDF ili unijeti tekst, te jedan od ponuđenih jezika. 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).

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

Kod 1. Maksimalna veličina učitane datoteke

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

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()  # Uzimanje teksta iz forme i brisanje viška praznih prostora
        if not text:
            return "No text provided"
```

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

- **Ekstrakcija teksta**

Metoda kojom se vrši ekstrakcija teksta - extract_text_from_pdf(pdf_file). Ova funkcija koristi biblioteku PyPDF2 za čitanje PDF datoteka. Otvara PDF datoteku, iterira kroz svaku stranicu, ekstraktuje tekst i vraća ga kao string.

```
import PyPDF2

def extract_text_from_pdf(pdf_file):
    text = ''  # Inicijalizuje prazan string za čuvanje ekstraktovanog teksta
    with pdf_file.stream as file:
        reader = PyPDF2.PdfReader(file)  # Učitava PDF datoteku
        for page in reader.pages:  # Iterira kroz sve stranice
            text += page.extract_text() + '\n'  # Ekstraktuje tekst
    return text  # Vraća kompletan tekst
```

Kod 3. Ekstrakcija teksta iz PDF-a

U kodu se tekst iz forme dobija putem request.form.get('text', '').strip(), gdje se višak praznih prostora uklanja.

- **Tokenizacija teksta/Obrada teksta**

Prije nego se pitanja generišu, potrebno je tokenizirati i obraditi tekst. Ovdje se koristi spaCy model (učitan ranije kao nlp) za obradu unesenog teksta. Funkcija nlp(text) tokenizuje tekst, što znači da razdvaja tekst na manje jedinice (tokene), kao što su riječi i interpunkcijski znakovi. Rezultat je objekat doc, koji sadrži sve informacije o tokenima.

Nakon tokenizacije, ovaj dio koda formira string processed_text koji sadrži tekst svih tokena, spojenih 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.

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

Kod 4. Tokenizacija i obrada teksta

- **Generisanje pitanja**

def generate_questions(text) - ova linija definiše funkciju generate_questions koja prima jedan argument, text. Ovaj argument predstavlja tekst iz kojeg će se generisati pitanja. Nakon toga se poziva prethodno definisani question_generator, koji je obično model za generisanje pitanja (npr. pretrenirani model iz biblioteke transformers).

- text: Ovo je ulazni tekst iz kojeg model generiše pitanja.
- max_length=50: Ova opcija postavlja maksimalnu dužinu generisanih pitanja na 50 tokena, što pomaže u kontroli dužine odgovora.
- num_beams=5: Ova opcija koristi "beam search" strategiju za generisanje pitanja. "Beam search"[2] je tehnika koja pretražuje više mogućih sekvenci tokom generisanja i bira najbolje na osnovu vjerovatnoće. U ovom slučaju, postavljanjem na 5, model će razmatrati 5 najboljih opcija.
- num_return_sequences=5: Ova opcija određuje koliko različitih pitanja će model generisati za dati tekst. U ovom slučaju, model će vratiti 5 generisanih pitanja.

return [question['generated_text'] for question in questions] - ova linija koristi list comprehension da izvuče generisana pitanja iz rezultata koji se vraćaju iz question_generator. Svako generisano pitanje se nalazi pod ključem 'generated_text', a ova linija formira listu tih pitanja i vraća je kao rezultat funkcije.

[2] Beam search je algoritam koji se koristi u mnogim modelima obrade prirodnog jezika i prepoznavanja govora kao završni sloj odlučivanja, koji odabire najbolji izlaz na osnovu ciljanih varijabli, poput maksimalne vjerovatnoće ili sljedećeg izlaznog karaktera. Payne, M. (2021). What is Beam Search? Explaining The Beam Search Algorithm | Width.ai. [online] www.width.ai. Dostupno na: https://www.width.ai/post/what-is-beam-search (pristupljeno 19. oktobra 2024)

Funkcija generate_questions (Kod 5) uzima tekst kao ulaz i koristi model za generisanje pitanja, postavljajući parametre za kontrolu kvaliteta i raznolikosti generisanih pitanja. Rezultat je lista generisanih pitanja koja se mogu dalje koristiti za obrazovne svrhe ili analizu.

```
def generate_questions(text):
    # Poziva question_generator sa zadatim tekstom i opcijama za generisanje pitanja
    questions = question_generator(
        text,  # Ulazni tekst iz kojeg se generišu pitanja
        max_length=50,  # Maksimalna dužina generisanih pitanja
        num_beams=5,  # Broj "beam search" pretraga za generisanje boljih pitanja
        num_return_sequences=5  # Broj generisanih pitanja koja će se vratiti
    )
```

Kod 5. Generisanje pitanja

- **Prikaz rezultata**

Rezultati se prikazuju u skladu sa definicijom u results.html (kod 6), 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 6. 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.

Osim toga, razvijeni kod pruža temelj za daljnji razvoj i unapređenje automatizovanih sistema u obrazovnom sektoru. Buduća istraživanja mogu se fokusirati na optimizaciju algoritama za generisanje pitanja, proširenje jezičkih modela, 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 takođe 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

- 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].

- Holdsworth, J. (2024). What Is Natural Language Processing? [online] IBM. Dostupno na: https://www.ibm.com/topics/natural-language-processing (pristupljeno: 18. oktobar 2024)

- Menzli, A. (2021). Tokenization in NLP: Types, Challenges, Examples, Tools. [online] neptune.ai. Dostupno na: https://neptune.ai/blog/tokenization-in-nlp (pristupljeno: 19. oktobar 2024)

- Payne, M. (2021). What is Beam Search? Explaining The Beam Search Algorithm | Width.ai. [online] www.width.ai. Dostupno na: https://www.width.ai/post/what-is-beam-search (pristupljeno 19. oktobra 2024)

- MyBib Contributors (2019). Harvard Referencing Generator – FREE – (updated for 2019). [online] MyBib. Dostupno na: https://www.mybib.com/tools/harvard-referencing-generator (pristupljeno: 19. oktobar 2024)