گزارش پروژه کارشناسی

پیش بینی جای‌خالی کلمات

استاد پروژه:

جناب آقای دکتر سید کمال الدین غیاثی شیرازی

نویسندگان:

محدثه جباری مقدم

فاطمه تبادکانی

گروه آموزشی:

کامپیوتر

پاییز ۱۴۰۰

تعهدنامه

اینجانب محدثه جباری مقدم و فاطمه تبادکانی دانشجوی دوره کارشناسی رشته مهندسی کامپیوتر دانشکده مهندسی دانشگاه فردوسی مشهد نویسنده پایان‌نامه پیش‌بینی جای‌خالی کلمات تحت راهنمایی دکتر غیاثی شیرازی متعهد می‌شوم :

- تحقیقات در این پایان نامه توسط اینجانبان انجام شده و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهش‌های محققان دیگر به مرجع مورد استفاده استناد شده است.
- مطالب مندرج در پایان نامه تاکنون توسط خود و یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ جا ارائه نشده است.
- کلیه حقوق معنوی این اثر متعلق به دانشگاه فردوسی مشهد می باشد و مقالات مستخرج با نام "دانشگاه فردوسی مشهد"و یا "Ferdowsi University of Mashhad"به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به دست آمدن نتایج اصلی پایان‌نامه تاثیرگذار بوده‌اند در مقالات مستخرج از رساله رعایت شده است.
- در کلیه مراحل انجام این پایان‌نامه، در مواردی که از موجود زنده (یا بافت‌های انها) استفاده شده است ضوابط و اصول اخلاقی رعایت شده است.
- در کلیه مراحل انجام این پایان‌نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته یا استفاده شده است، اصل رازداری، ضوابط و اصول اخلاق انسانی رعایت شده است .

مالکیت نتایج و حق نشر

- کلیه حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، کتاب، برنامه‌های رایانه‌ای، نرم‌افزارها و تجهیزات ساخته شده) متعلق به دانشگاه فردوسی مشهد می‌باشد. این مطلب باید به نحو مقتضی در تولیدات علمی مربوطه ذکر شود .

- استفاده از اطلاعات و نتایج موجود در پایان‌نامه بدون ذکر مرجع مجاز نمی‌باشد .

فهرست مطالب

معرفی پروژه

در این پروژه یک متن را به سیستم وارد می‌کنیم و سپس فعل های متن را بدست می‌آوریم و آن ها را به عنوان جای‌خالی جملات تعریف می‌کنیم.

از Mask-Language Modeling Bert استفاده می‌کنیم و جملات همراه با جای‌خالی را به مدل میدهیم و از Bert میخواهیم با توجه به جمله داده شده، جای خالی را برای ما پرکند. مدل برای هر جای‌خالی ۳۰۵۲۲ کلمه را پیشنهاد میدهد و با توجه به احتمال جواب‌هایی که Bert به ما برمیگرداند، جملات را سطح بندی می‌کنیم.

و در آخر یک اپلیکیشن می‌سازیم که در جهت تقویت و یادگیری زبان از آن می‌توان استفاده کرد به این صورت که جملات و جای‌خالی ها را به ما نشان میدهد و ما باید از بین گزینه های موجود، کلمه‌ی صحیح را انتخاب کنیم.

جای خالی میتواند برای صفات، قیدها یا اسم ها نیز به کار برود.

معرفی Bert

Bert یک روش مبتنی بر ترانسفومرها برای یادگیری بازنمایی زبان است.

این یک bidirectional transformer از پیش آموزش دیده است که به لطف دو رویکرد منحصر به فرد، mask language modeling(MLM)و next sentence prediction(NSP) از موفقیت بی نظیری در NLP برخوردار بوده است.

ما میتوانیم از بازنمایی های زبانی که BERT آموخته است برای کارهای مانند طبقه بندی متن و... استفاده کنیم تا به نتایج پیشرفته ای در مورد مسئله خود دست یابیم.

در بسیاری از موارد، ممکن است بتوانیم مدل bert را که از قبل آموزش داده شده است، خارج کنیم و آن را با توجه به مدل مورد نیاز در مسئله پیاده سازی کنیم.

در این پروژه باتوجه به مسئله fine-tune هم انجام دادیم اما چون دادگان ما انگلیسی است تغییرات چندانی در نتیجه حاصل نشد.

این ترنسفورمر که مورد توجه بسیاری از افراد قرار گرفت.

ما در واقع از Bert اینگونه استفاده می کنیم که یک جمله ناقص وارد می کنیم و از BERT می خواهیم که جمله را برای ما کامل کند.

به عنوان مثال در جمله زیر می خواهیم جای‌خالی را پر کنیم :

In Autumn the -------- fall from the trees.

در این جمله به احتمال زیاد جواب جای خالی را میدانید و این به این دلیل است که متن جمله را در نظر گرفته اید و مفهوم جمله را متوجه شده اید.

چیزهای زیادی از درختان می‌افتند به عنوان مثال بلوط‌ها، شاخه‌ها، برگ‌ها. اما ما در پاییز شرایط دیگری داریم، که جستجوی ما را محدود می‌کند، محتمل‌ترین چیزی که در پاییز از درخت می‌افتد، برگ است.

Answer: leaves

ما به عنوان انسان ترکیبی از دانش عمومی و درک زبانی برای رسیدن به این نتیجه استفاده می کنیم. برای BERT این حدس از مطالعه زیاد و یادگیری الگوهای زبانی فوق العاده خوب حاصل می شود.

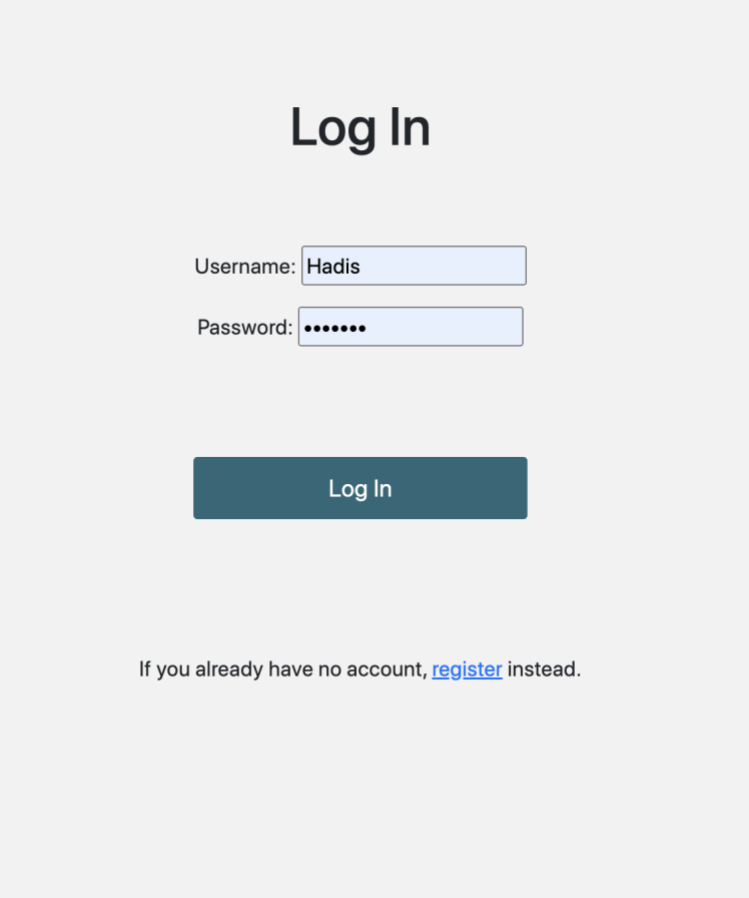
BERT ممکن است نداند پاییز، درختان و برگ ها چیست، اما می داند که با توجه به الگوهای زبانی و زمینه این کلمات، پاسخ به احتمال زیاد برگ ها است.

ما در پروژه یک متن از کتاب Actual test آیلتس را انتخاب کرده ایم.

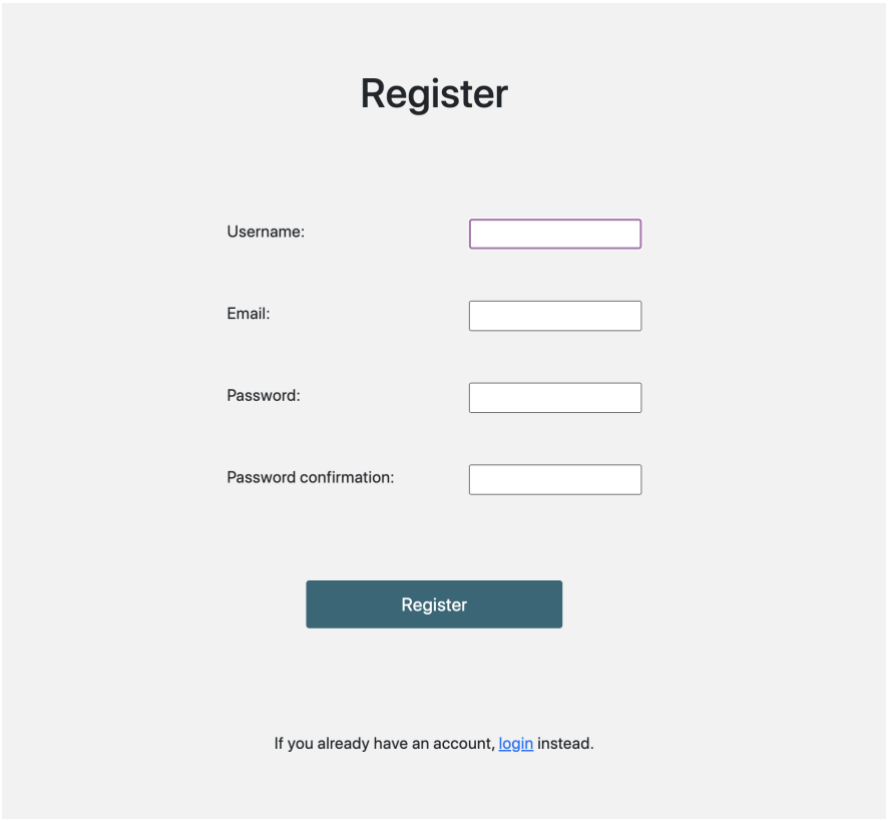
جای خالی ها را نیز در سه سطح بررسی کردیم :Basic, Intermediate, Advanced

نشان دادن صفحات سایت

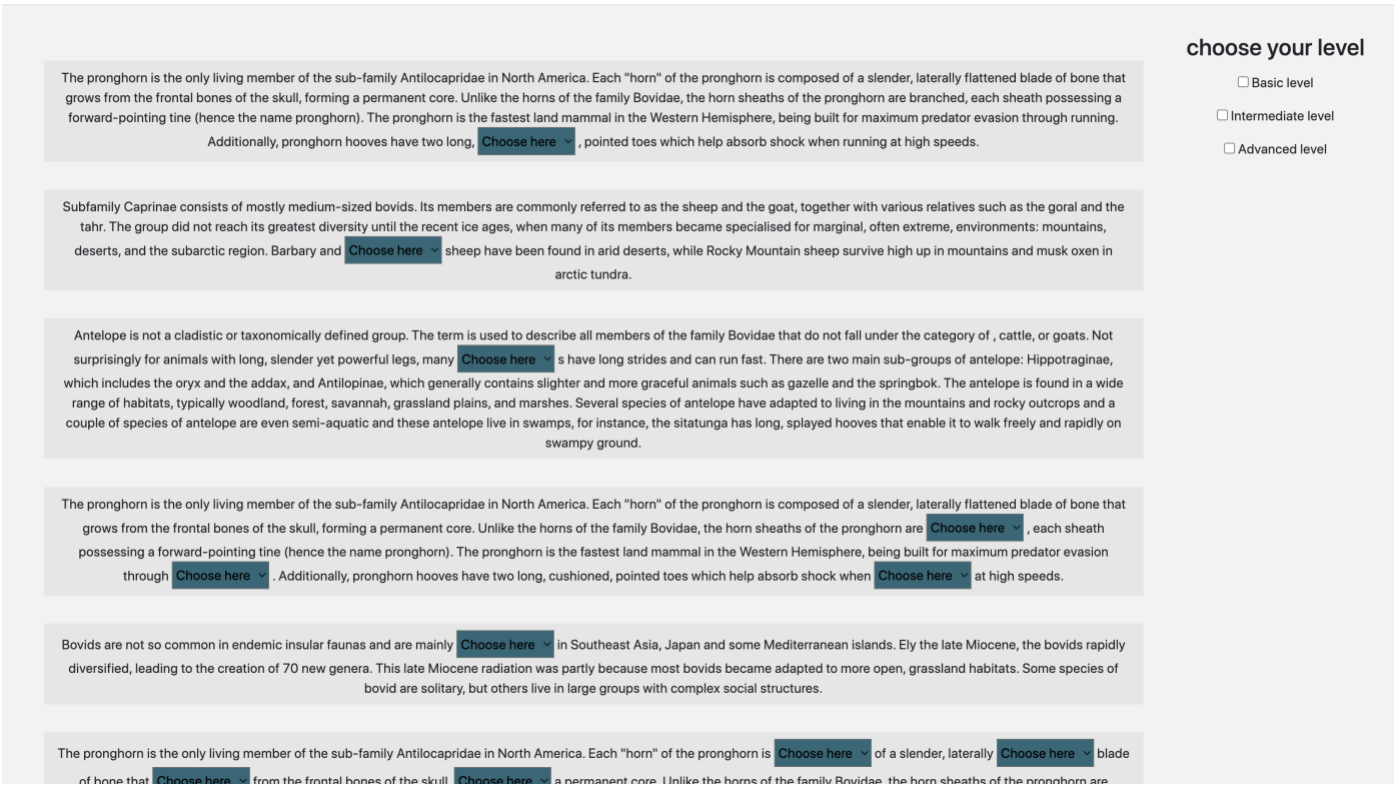
صفحه‌ی ورود:



صفحه‌ی ثبت نام:

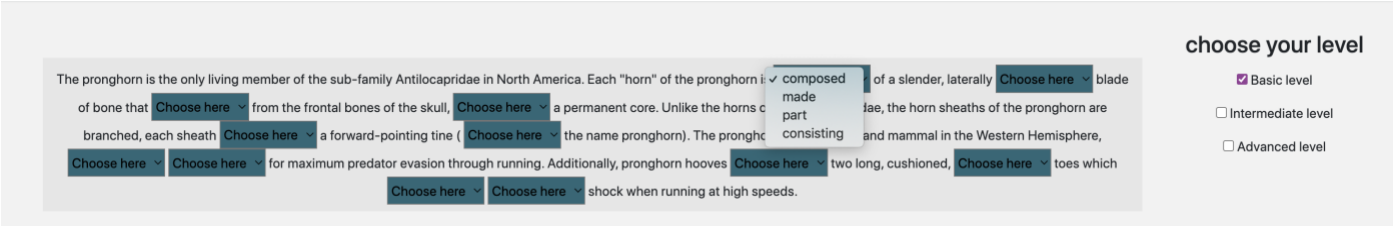


اولین صفحه ی ورود کاربر:

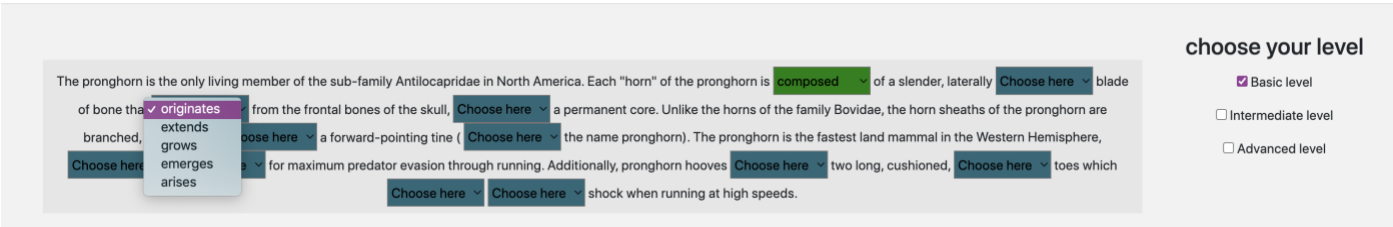


اگر کاربر سطح Basic را انتخاب کند:

کاربر روی اولین جای خالی کلیک میکند:

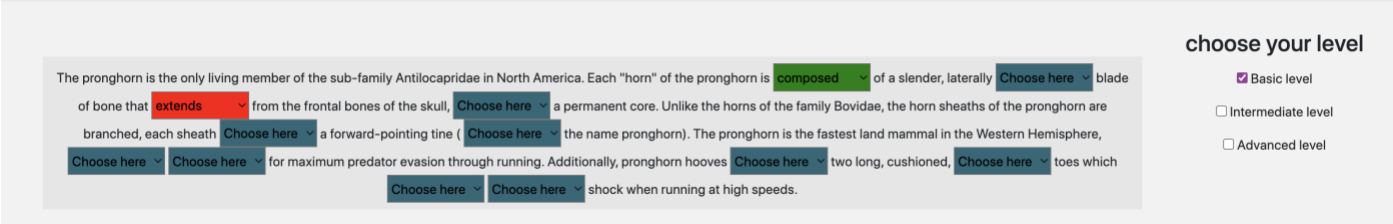


گزینه ها را مشاهده می‌کنیم و گزینه‌ی مناسب انتخاب می‌شود.

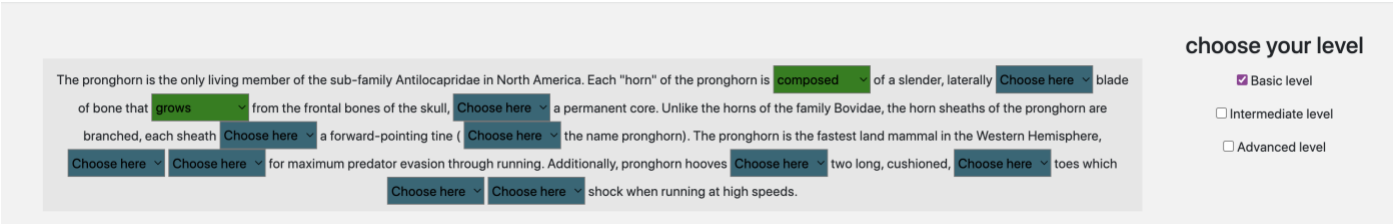


اگر کلمه ای که انتخاب می‌شود، درست باشد به رنگ سبز در می‌آید در غیر این صورت به رنگ قرمز در می‌اید.

یک تصویر بالا کلمات یک جای‌خالی دیگر را نمایش داده‌ایم.

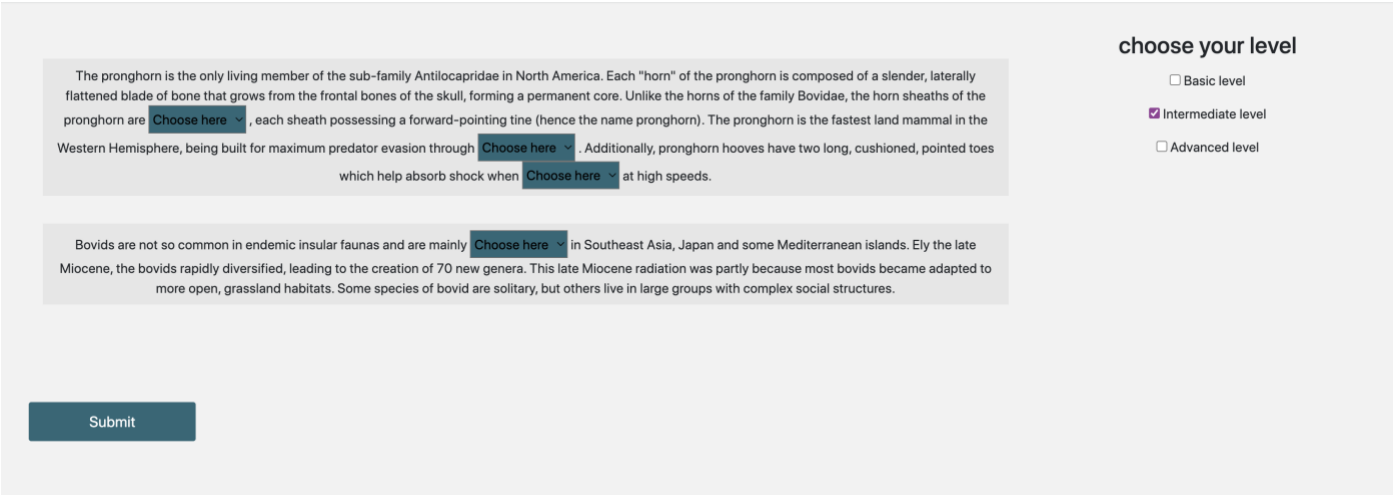


کلمه‌ی انتخابی دوم صحیح نمی‌باشد.

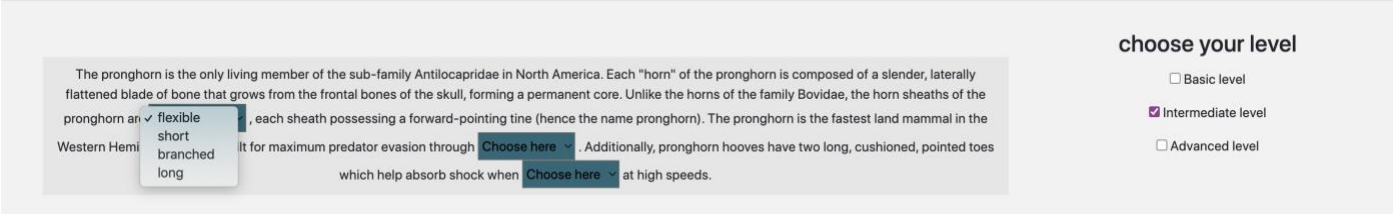


کلمه‌ی grows صحیح می‌باشد.

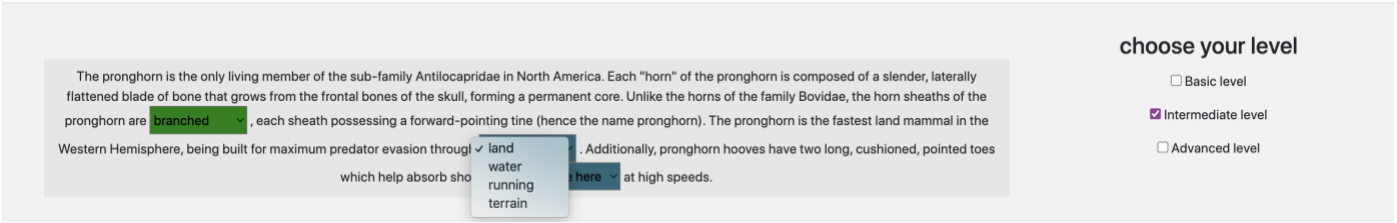
اگر کاربر سطح Intermediate را انتخاب کند:

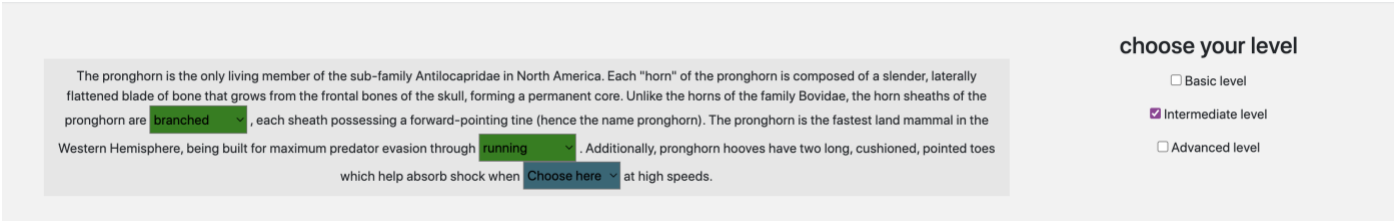


کلمات برای اولین جای‌خالی نمایش داده می‌شود:



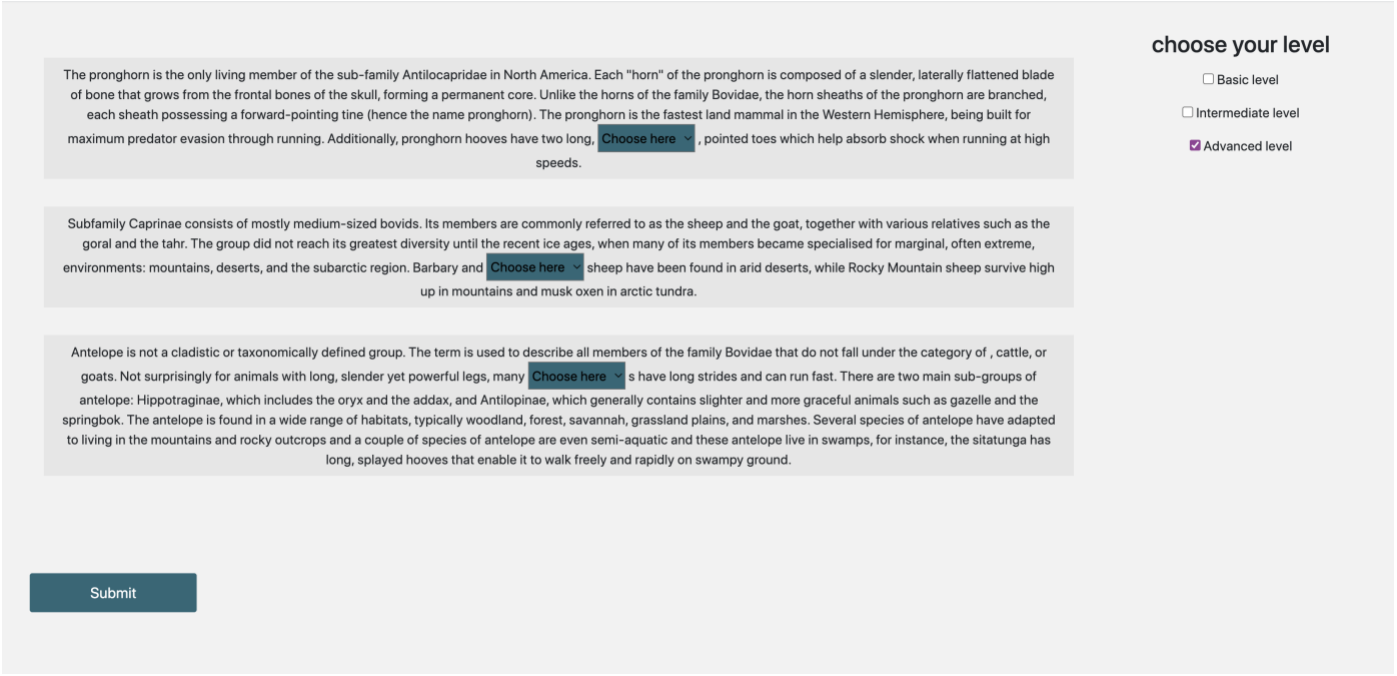
کلمات برای دومین جای‌خالی نمایش داده می‌شود:



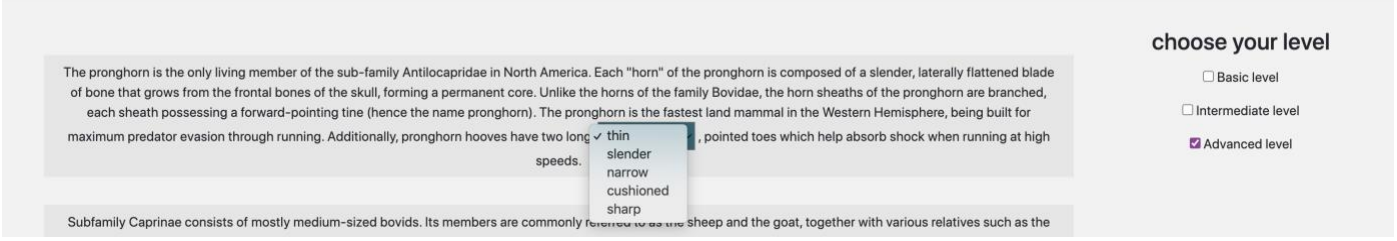


کلمات صحیح را مشاهده می‌کنیم.

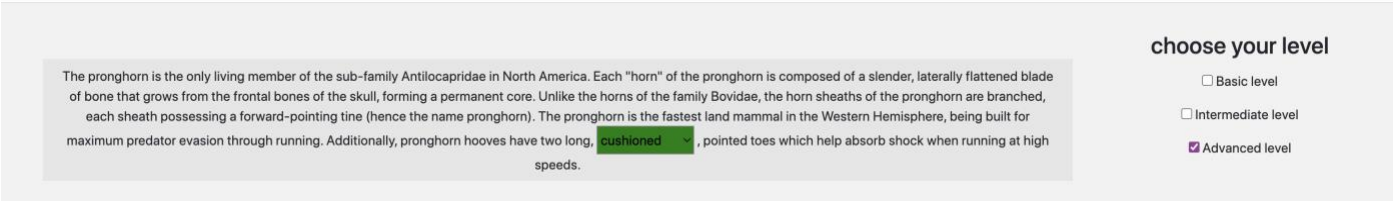
اگر کاربر سطح Advanced را انتخاب کند:



گزینه های مربوط به جای‌خالی را می‌بینیم:



گزینه‌ی صحیح نمایش داده می‌شود:



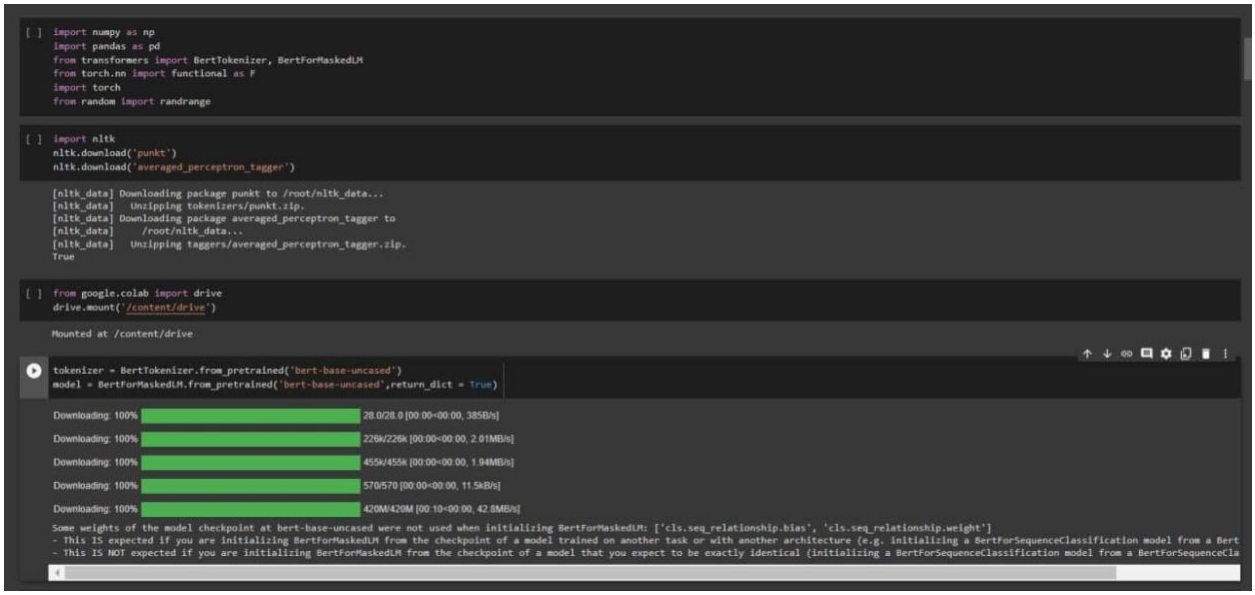
کد پیش‌بینی جای‌خالی کلمات:

برای بخش هوش مصنوعی این پروژه که درواقع نوعی مسئله NLP(Natural Language Processing) است, همان طور که گفته شده از Bert استفاده کردیم. کد مربوط به این بخش را در ادامه آورده شده و هر بخش توضیح داده می شود:

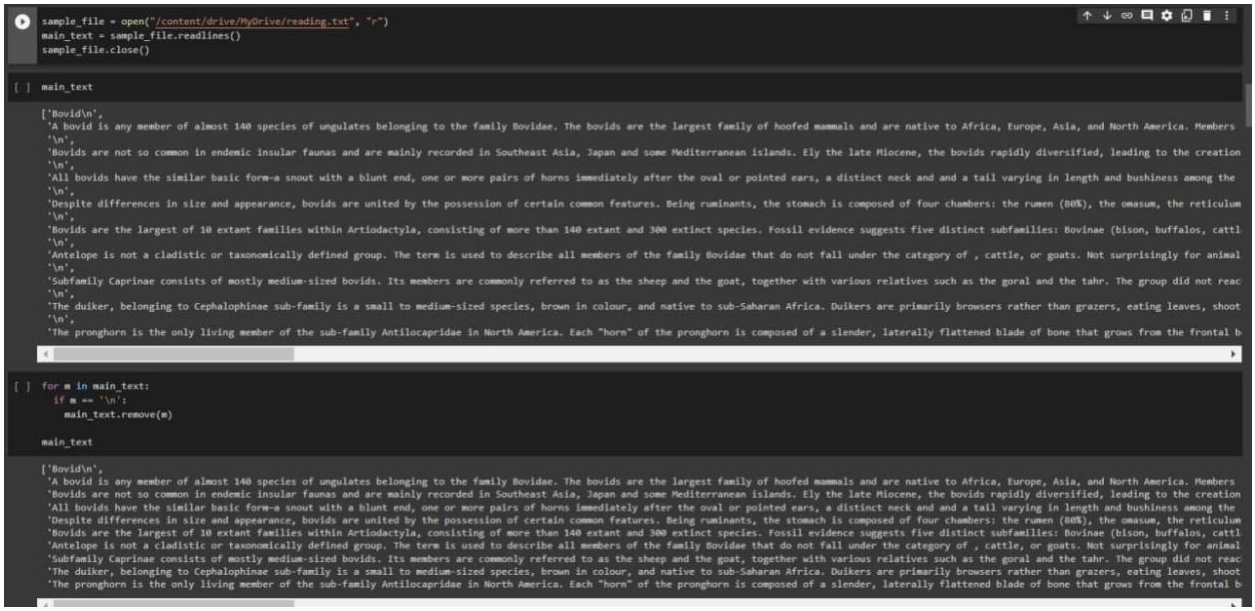
در این قسمت برای استفاده از کتابخانه های مورد نظر transformers را نصب می کنیم:

```
!pip install transformers
Collecting transformers
  Downloading transformers-4.16.2-py3-none-any.whl (3.5 MB)
     |████████████████████████████████| 3.5 MB 5.2 MB/s
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from transformers) (21.3)
Collecting huggingface-hub<1.0,>=0.1.0
  Downloading huggingface_hub-0.4.0-py3-none-any.whl (67 kB)
     |████████████████████████████████| 67 kB 4.5 MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from transformers) (2.23.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from transformers) (4.62.3)
Collecting sacremoses
  Downloading sacremoses-0.0.47-py2.py3-none-any.whl (895 kB)
     |████████████████████████████████| 895 kB 54.7 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from transformers) (3.4.2)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (1.21.5)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (2019.12.20)
Collecting tokenizers!=0.11.3,>=0.10.1
  Downloading tokenizers-0.11.5-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (6.8 MB)
     |████████████████████████████████| 6.8 MB 30.0 MB/s
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from transformers) (4.11.0)
Collecting pyyaml>=5.1
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (596 kB)
     |████████████████████████████████| 596 kB 62.2 MB/s
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-packages (from huggingface-hub<1.0,>=0.1.0->transformers) (3.10.0.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>=20.0->transformers) (3.0.7)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->transformers) (3.7.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (2021.10.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers) (1.1.0)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers) (7.1.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers) (1.15.0)
Installing collected packages: pyyaml, tokenizers, sacremoses, huggingface-hub, transformers
  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed huggingface-hub-0.4.0 pyyaml-6.0 sacremoses-0.0.47 tokenizers-0.11.5 transformers-4.16.2
```

سپس کتابخانه های دیگری که نیاز است را import می کنیم و نسخه pretrain شده bert را دانلود می کنیم:



به عنوان متن ورودی, از فایل reading.txt استفاده می کنیم. محتویات این فایل را در متغیر main_text ریخته و "\n" را از آن حذف می کنیم:



در مرحله بعد در بین جملات موجود به دنبال کلماتی می گردیم که برای جاخالی مناسب‌تر باشند که برای این کار در هر جمله فعل های آن را انتخاب می کنیم. به این منظور با استفاده از کتابخانه nltk پس از tokenize کردن هر جمله کلماتی که توسط این کتابخانه verb تشخیص داده می‌شوند را انتخاب کرده و در آرایه verbs ذخیره می‌کنیم. همچنین با توجه به اینکه افعالی مثل is و are که تعداد حروف کمی دارند معمولا کلمات آسانی بوده و به عنوان جاخالی مناسب نیستند, افعال با طول کمتر از 3 را در نظر نمی گیریم:

```
[ ]  verbs = [[] for i in range(length)]

     for i in range(length):
       text = nltk.word_tokenize(main_text[i]);
       pos_tagged = nltk.pos_tag(text)
       for ele in pos_tagged:
         if(ele[1].startswith('VB') and len(ele[0])>3):
           verbs[i].append(ele[0])
```

سپس برای اینکه جاخالی های یک جمله کلمات تکراری نباشند با استفاده از کد زیر کلمات تکراری را از ارایه verbs حذف می کنیم:

```
   new_verbs = [[] for i in range(length)]

   for i in range(length):
     new_verbs[i] = list(dict.fromkeys(verbs[i]))
```

در قسمت اصلی کد با استفاده از bert, کلماتی که به عنوان جاخالی انتخاب شده اند را در نظر گرفته و کلماتی که Bert برای آن جاخالی ها مناسب می داند را در نظر می گیریم. Bert یک دیکشنری دارد که شامل 30552 کلمه است. پس از ساخت مدل با استفاده از برت، نتیجه کلمات در output ریخته می شود. سپس برای هر کلمه یک احتمال به دست می‌اورد که در logits ذخیره میشود، این اعداد استاندارد نبوده و ممکن است منفی یا اعشاری یا ... باشند. برای استانداردسازی آن ها از Softmax استفاده می‌کنیم:

```
for i in range(length):

  for verb in new_verbs[i]:

    labels_one_sentence1 = []
    labels_one_sentence2 = []
    labels_one_sentence3 = []

    split_text = main_text[i].split(verb, 1)
    text1 = split_text[0] + tokenizer.mask_token + split_text[1]

    input = tokenizer.encode_plus(text1, return_tensors = "pt")
    mask_index = torch.where(input["input_ids"][0] == tokenizer.mask_token_id)

    output = model(**input)
    logits = output.logits

    softmax = F.softmax(logits, dim = -1)
    mask_word = softmax[0, mask_index, :]
```

در مرحله بعد کلمات جاخالی را به سه سطح سطح بندی می کنیم. برای این کار باید ببینیم کلمه ای که باید در جای خالی قرار بگیرد, توسط برت به عنوان چندمین کلمه انتخاب می‌شود. اگر جزء کلمات اول تا ۱۵ ام باشد در سطح ۱ (Basic), اگر بین 15 و 255 امین کلمه باشد سطح 2 (Intermediate) و اولویت آن بیشتر از 256 باشد سطح 3 (Advanced) قرار میگیرد:

```
level1 = top_10[0:(4**2)-1]
level2 = top_10[4**2:(4**4)-1]
level3 = top_10[(4**4):(4**6)-1]

#level_1
if tokenizer.encode_plus(verb).input_ids[1] in level1:
  answers1[i].append(verb)
  labels_one_sentence1.append(verb)
  options = torch.topk(mask_word, 4, dim = 1)[1][0]
  for token in options:
    word = tokenizer.decode([token])
    if (word!=verb):
      labels_one_sentence1.append(word)
  index = labels_one_sentence1.index(verb)
  correct_index = randrange(4)
  replace(labels_one_sentence1, correct_index, index)
  labels1[i].append(labels_one_sentence1)
#level_2
elif tokenizer.encode_plus(verb).input_ids[1] in level2:
  answers2[i].append(verb)
  labels_one_sentence2.append(verb)
  options = torch.topk(mask_word, 3, dim = 1)[1][0]
  for token in options:
    word = tokenizer.decode([token])
    labels_one_sentence2.append(word)
  correct_index = randrange(4)
  replace(labels_one_sentence2, correct_index, 0)
  labels2[i].append(labels_one_sentence2)
#level_3
elif tokenizer.encode_plus(verb).input_ids[1] in level3:
  answers3[i].append(verb)
  labels_one_sentence3.append(verb)
  options = torch.topk(mask_word, 4, dim = 1)[1][0]
  for token in options:
    word = tokenizer.decode([token])
    labels_one_sentence3.append(word)
  correct_index = randrange(4)
  replace(labels_one_sentence3, correct_index, 0)
  labels3[i].append(labels_one_sentence3)
```

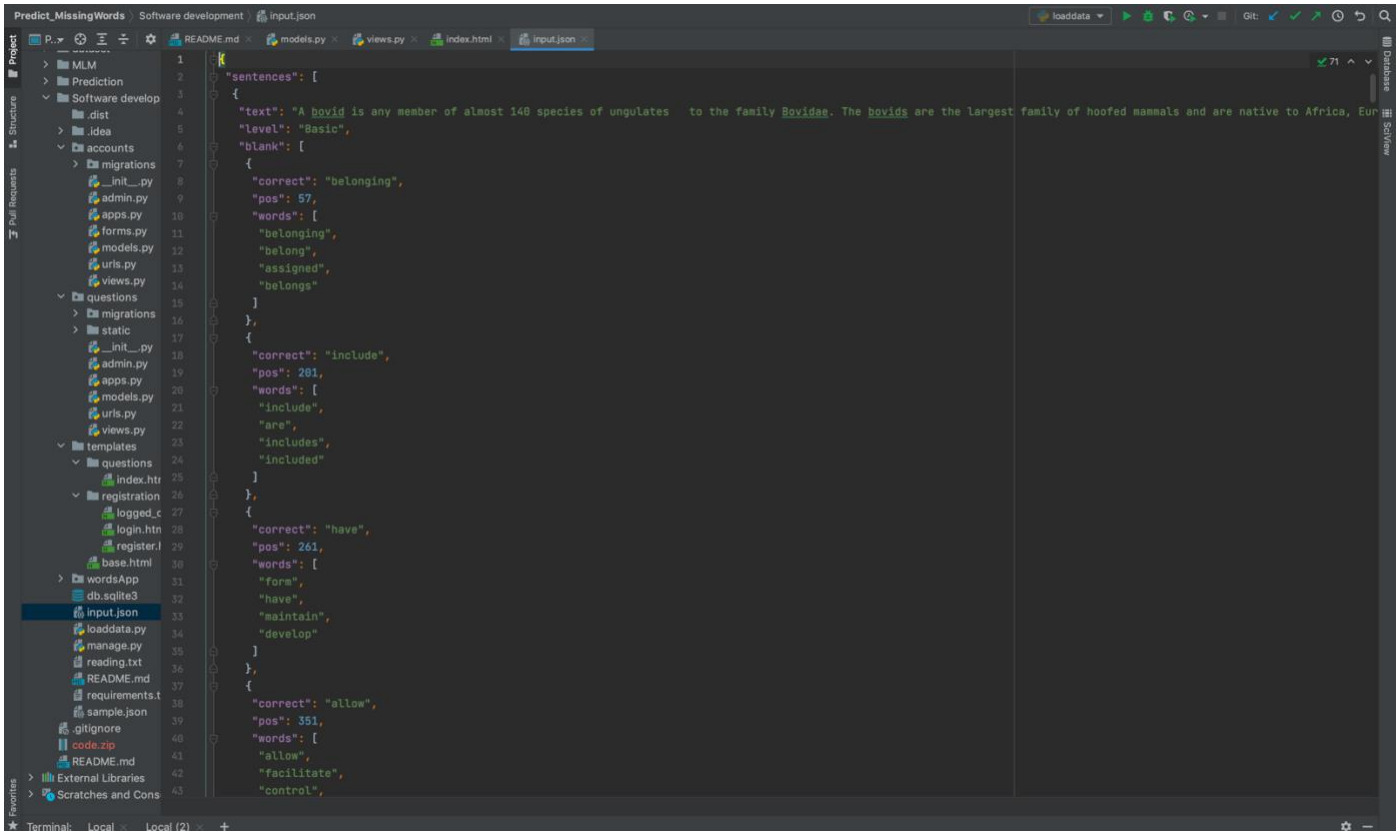
خروجی را در یک dictionary ذخیره می کنیم. فرم آن نیز به این صورت است که برای هر جمله یک دیکشنری ذخیره میشود که شامل text یعنی جمله اصلی, level یعنی سطح آن و آرایه blank که شامل جای‌خالی هاست, می باشد:

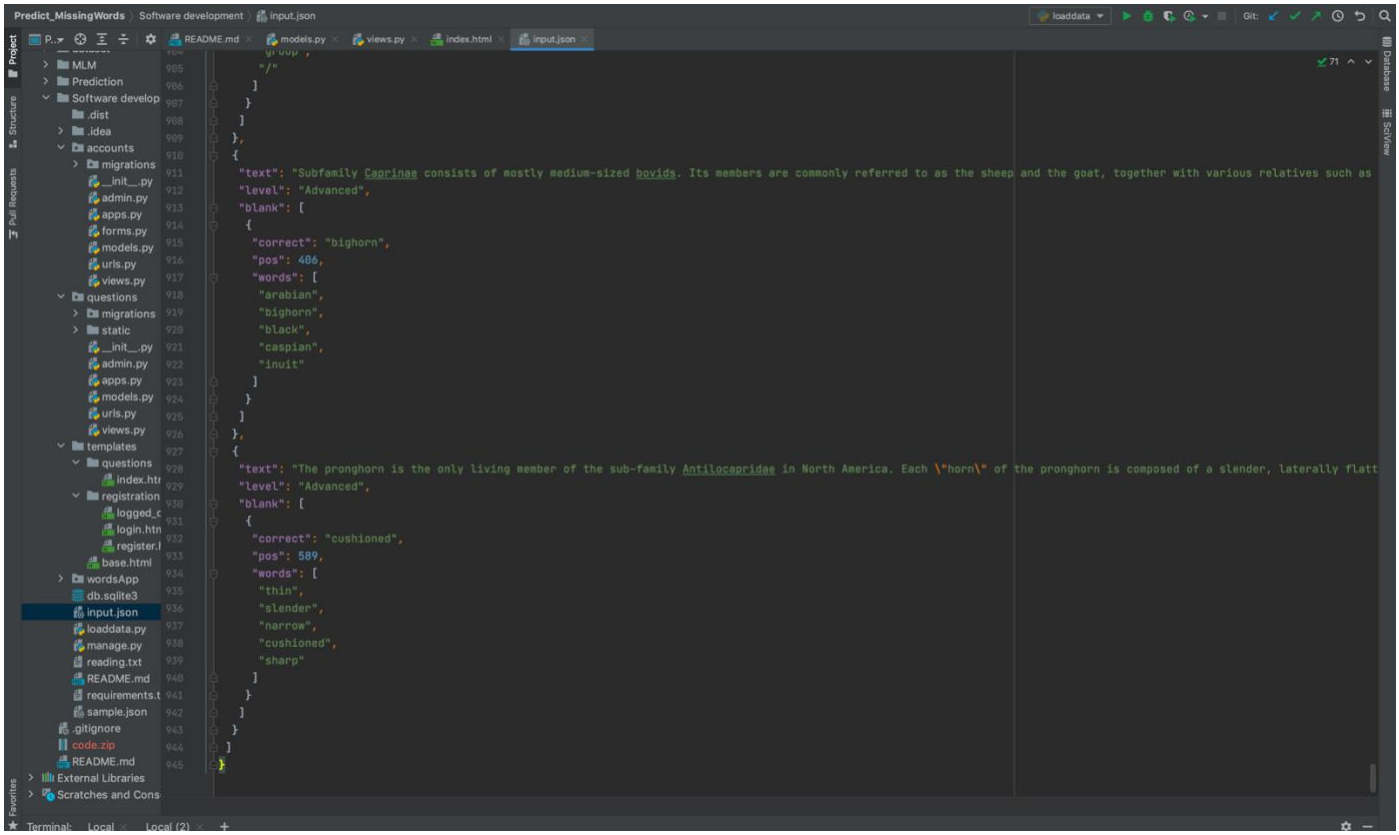
```
dictionary = {"sentences": [{} for i in range(length*3)]}

for i in range(length):
  #level_1
  if(len(answers1[i]) != 0):
    dictionary["sentences"][i].update({"text": main_text[i]})
    dictionary["sentences"][i].update({"level": "Basic"})
    dictionary["sentences"][i].update({"blank": [{} for j in range(len(answers1[i]))]})
    for j in range(len(answers1[i])):
      dictionary["sentences"][i]["blank"][j].update({"correct": answers1[i][j]})
      pos = dictionary["sentences"][i]["text"].index(answers1[i][j])
      dictionary["sentences"][i]["blank"][j].update({"pos": pos})
      dictionary["sentences"][i]["blank"][j].update({"words": labels1[i][j]})
      dictionary["sentences"][i].update({"text": dictionary["sentences"][i]["text"].replace(answers1[i][j], " ", 1)})
  #level_2
  if(len(answers2[i]) != 0):
    k = i+length
    dictionary["sentences"][k].update({"text": main_text[i]})
    dictionary["sentences"][k].update({"level": "Intermediate"})
    dictionary["sentences"][k].update({"blank": [{} for j in range(len(answers2[i]))]})
    for j in range(len(answers2[i])):
      dictionary["sentences"][k]["blank"][j].update({"correct": answers2[i][j]})
      pos = dictionary["sentences"][k]["text"].index(answers2[i][j])
      dictionary["sentences"][k]["blank"][j].update({"pos": pos})
      dictionary["sentences"][k]["blank"][j].update({"words": labels2[i][j]})
      dictionary["sentences"][k].update({"text": dictionary["sentences"][k]["text"].replace(answers2[i][j], " ", 1)})
  #level_3
  if(len(answers3[i]) != 0):
    k = i + length*2
    dictionary["sentences"][k].update({"text": main_text[i]})
    dictionary["sentences"][k].update({"level": "Advanced"})
    dictionary["sentences"][k].update({"blank": [{} for j in range(len(answers3[i]))]})
    for j in range(len(answers3[i])):
      dictionary["sentences"][k]["blank"][j].update({"correct": answers3[i][j]})
      pos = dictionary["sentences"][k]["text"].index(answers3[i][j])
      dictionary["sentences"][k]["blank"][j].update({"pos": pos})
```

سپس این دیکشنری را در یک فایل json ذخیره می کنیم که این فایل ورودی فایل جنگوی ما خواهد بود:

```
import json
json_object = json.dumps(dictionary, indent=len(dictionary))

# Writing to input.json
with open("input.json", "w") as outfile:
    outfile.write(json_object)
```

بخشی از فایل json:


Predict_MissingWords
Software development
input.json
loaddata
README.md
models.py
views.py
index.html
input.json
{
"sentences": [
{
"text": "A bovid is any member of almost 140 species of ungulates to the family Bovidae. The bovids are the largest family of hoofed mammals and are native to Africa, Eur
"level": "Basic",
"blank": [
{
"correct": "belonging",
"pos": 57,
"words": [
"belonging",
"belong",
"assigned",
"belongs"
]
},
{
"correct": "include",
"pos": 201,
"words": [
"include",
"are",
"includes",
"included"
]
},
{
"correct": "have",
"pos": 261,
"words": [
"form",
"have",
"maintain",
"develop"
]
},
{
"correct": "allow",
"pos": 351,
"words": [
"allow",
"facilitate",
"control",


Predict_MissingWords
Software development
input.json
loaddata
README.md
models.py
views.py
index.html
input.json
"/*"
]
}
]
},
{
"text": "Subfamily Caprinae consists of mostly medium-sized bovids. Its members are commonly referred to as the sheep and the goat, together with various relatives such as
"level": "Advanced",
"blank": [
{
"correct": "bighorn",
"pos": 406,
"words": [
"arabian",
"bighorn",
"black",
"caspian",
"inuit"
]
}
]
},
{
"text": "The pronghorn is the only living member of the sub-family Antilocapridae in North America. Each \"horn\" of the pronghorn is composed of a slender, laterally flatt
"level": "Advanced",
"blank": [
{
"correct": "cushioned",
"pos": 589,
"words": [
"thin",
"slender",
"narrow",
"cushioned",
"sharp"
]
}
]
}
]
}

توسعه نرم افزاری:

در این پروژه برای توسعه نرم افزاری از پایتون استفاده شده است که برای back-end از جنگو و برای front-end از template engine جنگو استفاده کرده ایم.

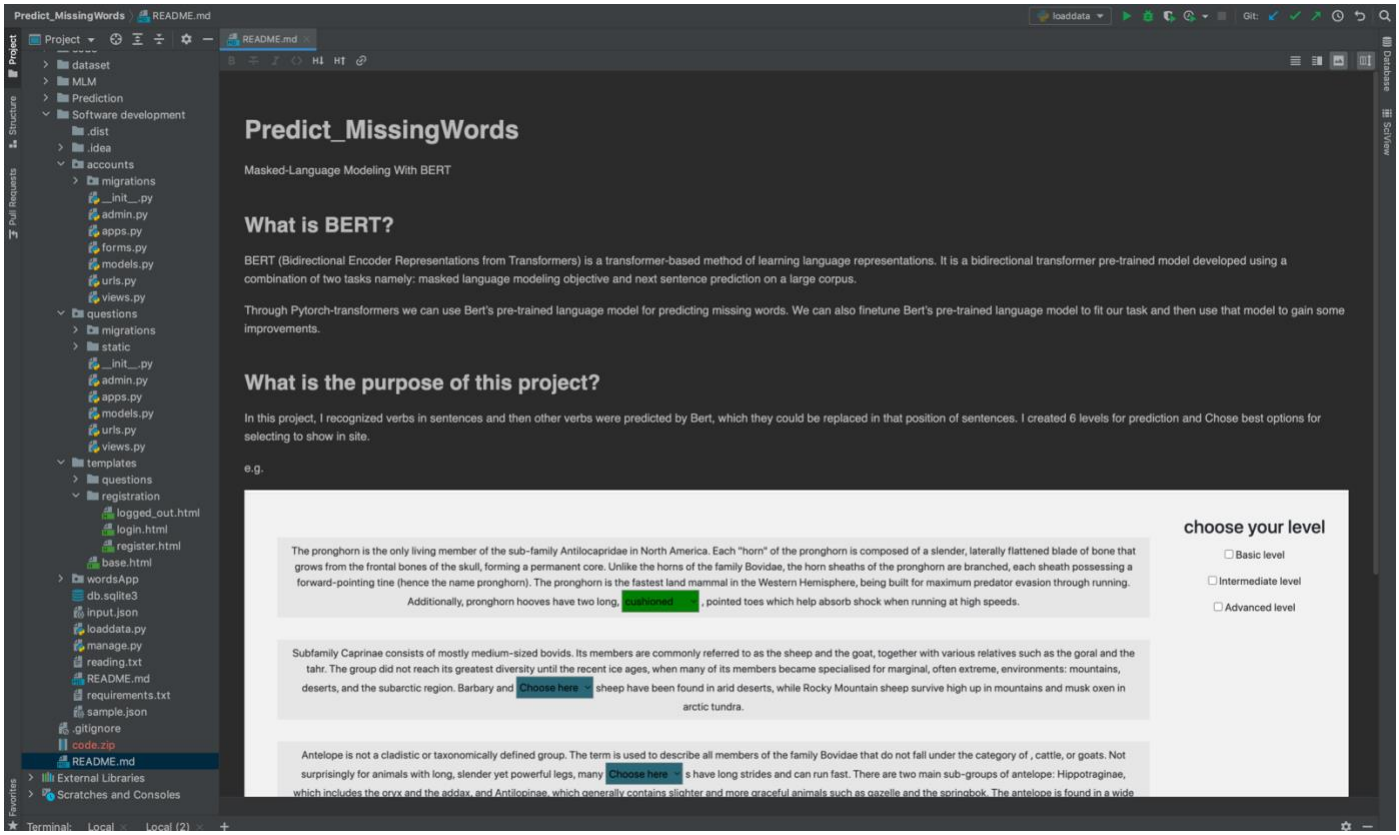
چرا جنگو را انتخاب کردیم؟

جنگو (Django)یک چارچوب نرم افزاری تحت وب آزاد و متن باز است که به زبان پایتون نوشته شده‌است و از معماری مدل-نما-قالب (Model–View–Template)پیروی می‌کند. هدف اصلی جنگو ساخت آسان سایت‌های پیچیده و وابسته به دیتابیس است و بر پایه‌ی قابلیت استفاده‌ی مجدد و **قابل اتصال بودن** اجزای مختلف، توسعه‌ی سریع و اصل عدم تکرارطراحی شده‌است. جنگو سراسر از پایتون استفاده می‌کند، حتی برای تنظیمات، فایل‌ها و مدل‌های اطلاعات.

باتوجه به قابلیت هایی که جنگو در اختیار ما قرار می‌دهد، تصمیم گرفتیم که از پایتون استفاده کنیم.

نمای کلی از ساختار پروژه:

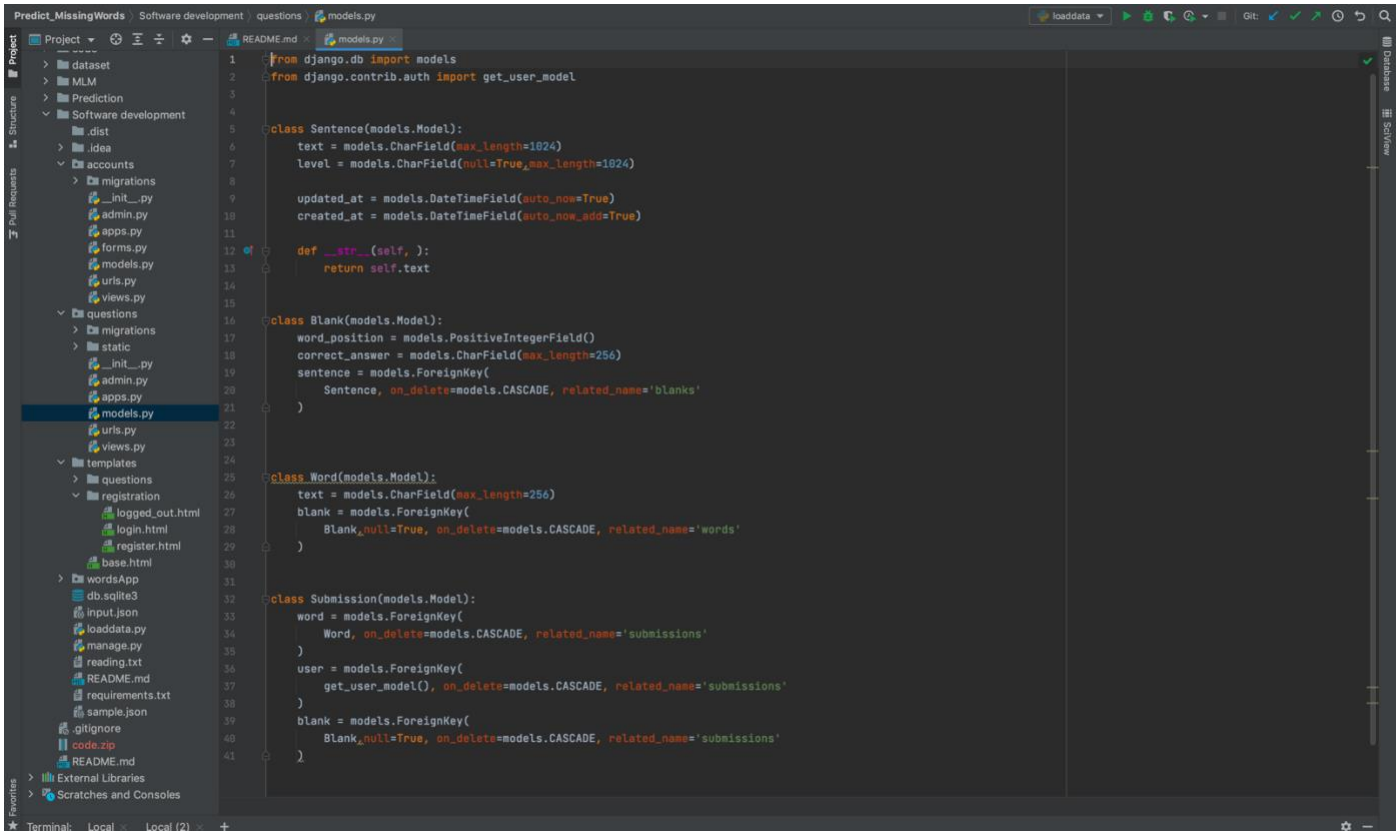
در این صفحه app های مربوط به پروژه رو می بینید که شامل کاربران و سوالات میشود. صفحات manage.py برای ران کردن سرور است و همچنین باقی صفحات را مشاهده میکنید.



مدل های دیتابیس:

ما در بخش questions، در فایل models.py دیتابیس را طراحی می‌کنیم. دیتابیس ما یک مدل اصلی به اسم sentence دارد که شامل متن و سطح جمله می‌باشد. هر جمله می تواند چندین جای خالی داشته باشد که ما ان را با نام Blank مشخص کرده ایم و هر جای خالی میتواند چندتا کلمه داشته باشد که با نام word تعریف شده است.

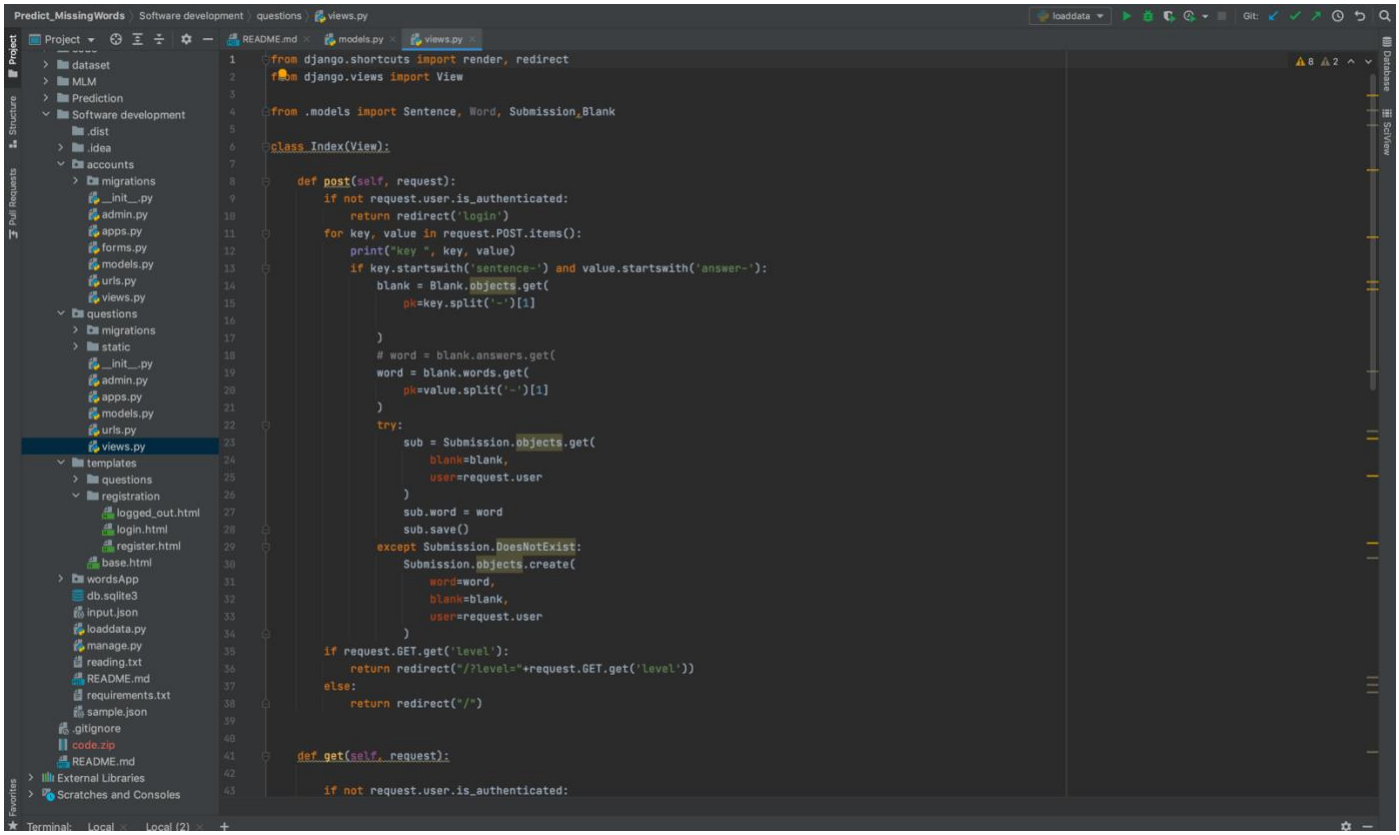
هر کاربر برای هر جای‌خالی میتواند یک گزینه را انتخاب کند که با توجه به همین مسئله اطلاعات کاربر در دیتابیس ذخیره می‌شود.



فایل View:

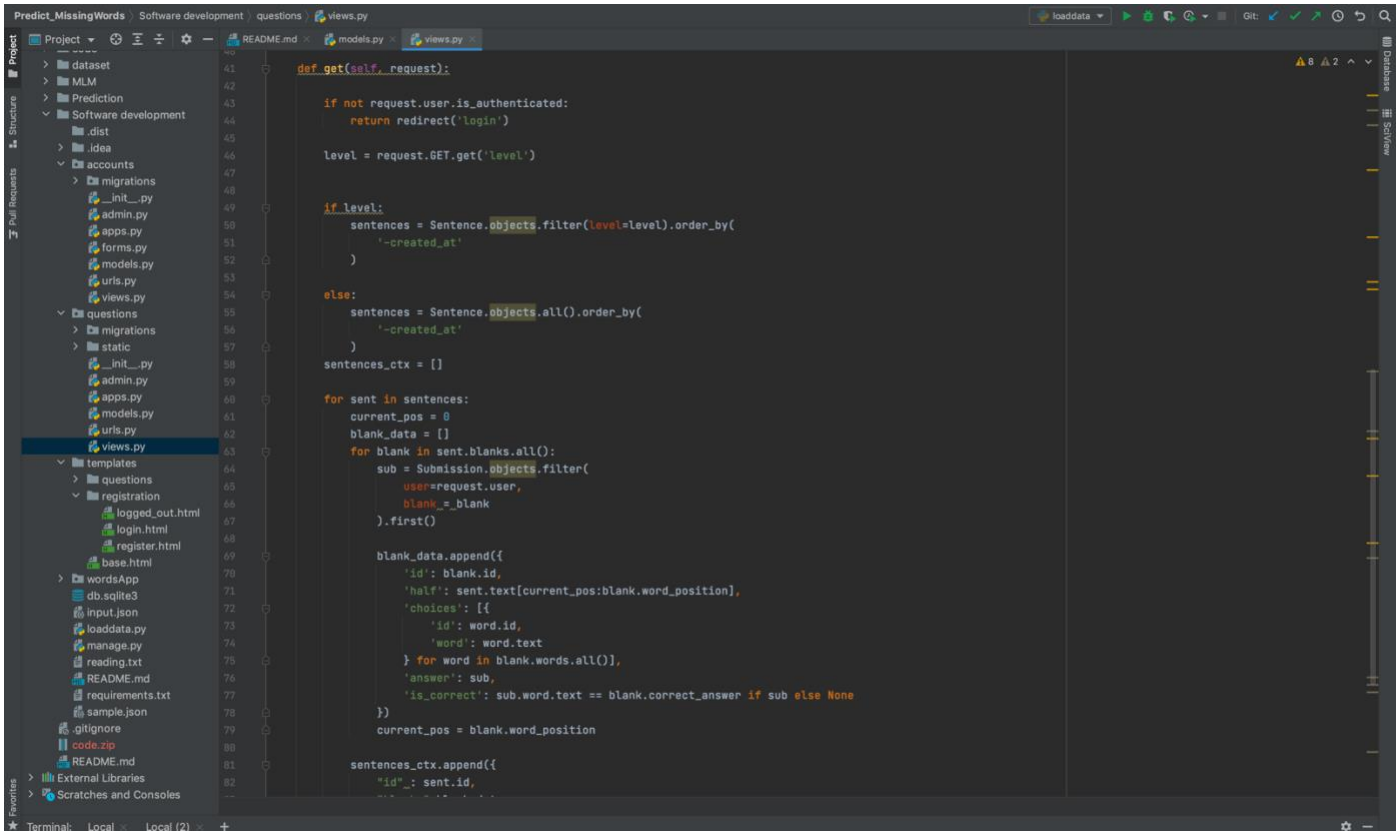
در جنگو بخش کنترل کردن کار در فایل view اتفاق می‌افتد.

در متد post ما submission را انجام می‌دهیم. با توجه به کلمه‌ای که کاربر وارد کرده و جای خالی که انتخاب کرده، اطلاعاتش در دیتابیس ذخیره می شود.

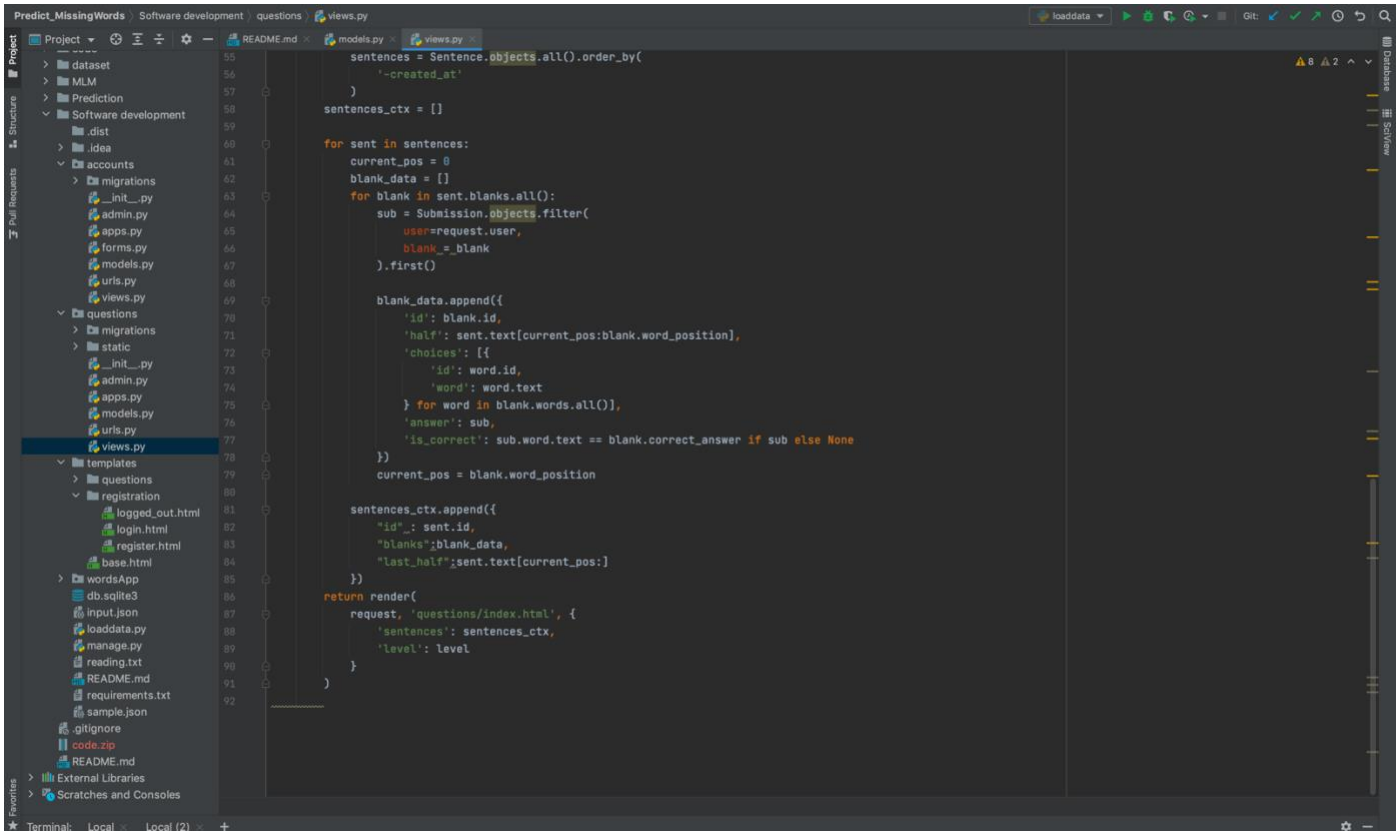



```python
from django.shortcuts import render, redirect
from django.views import View

from .models import Sentence, Word, Submission,Blank

class Index(View):

    def post(self, request):
        if not request.user.is_authenticated:
            return redirect('login')
        for key, value in request.POST.items():
            print("key ", key, value)
            if key.startswith('sentence-') and value.startswith('answer-'):
                blank = Blank.objects.get(
                    pk=key.split('-')[1]

                )
                # word = blank.answers.get(
                word = blank.words.get(
                    pk=value.split('-')[1]
                )
                try:
                    sub = Submission.objects.get(
                        blank=blank,
                        user=request.user
                    )
                    sub.word = word
                    sub.save()
                except Submission.DoesNotExist:
                    Submission.objects.create(
                        word=word,
                        blank=blank,
                        user=request.user
                    )
        if request.GET.get('level'):
            return redirect("/?level="+request.GET.get('level'))
        else:
            return redirect("/")


    def get(self, request):

        if not request.user.is_authenticated:
```

در متد get کوئری های لازم زده میشود تا دیتای مورد نظر ما بدست آید و این دیتا را می‌گیریم و به template engine جنگو میدهیم که html را میگیرد و داخل آن، دیتا را قرار میدهد و به کاربر نشان میدهد.



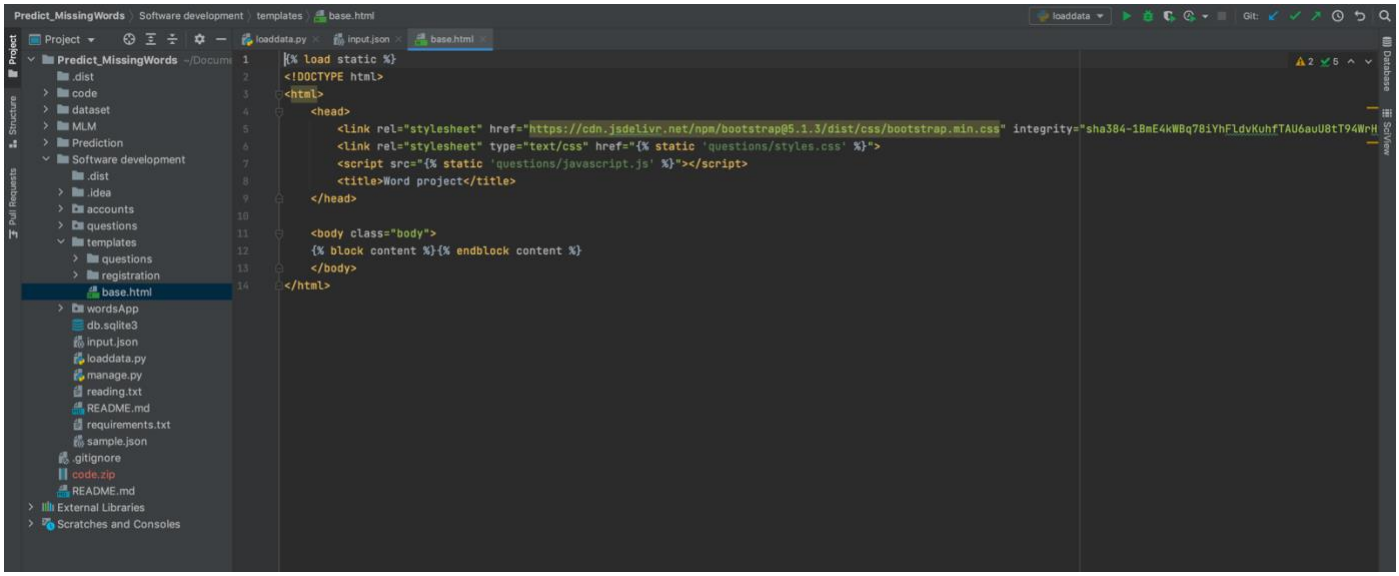
```
def get(self, request):

    if not request.user.is_authenticated:
        return redirect('login')

    level = request.GET.get('level')


    if level:
        sentences = Sentence.objects.filter(level=level).order_by(
            '-created_at'
        )

    else:
        sentences = Sentence.objects.all().order_by(
            '-created_at'
        )
    sentences_ctx = []

    for sent in sentences:
        current_pos = 0
        blank_data = []
        for blank in sent.blanks.all():
            sub = Submission.objects.filter(
                user=request.user,
                blank_=_blank
            ).first()

            blank_data.append({
                'id': blank.id,
                'half': sent.text[current_pos:blank.word_position],
                'choices': [{
                    'id': word.id,
                    'word': word.text
                } for word in blank.words.all()],
                'answer': sub,
                'is_correct': sub.word.text == blank.correct_answer if sub else None
            })
            current_pos = blank.word_position

        sentences_ctx.append({
            "id"_: sent.id,
```



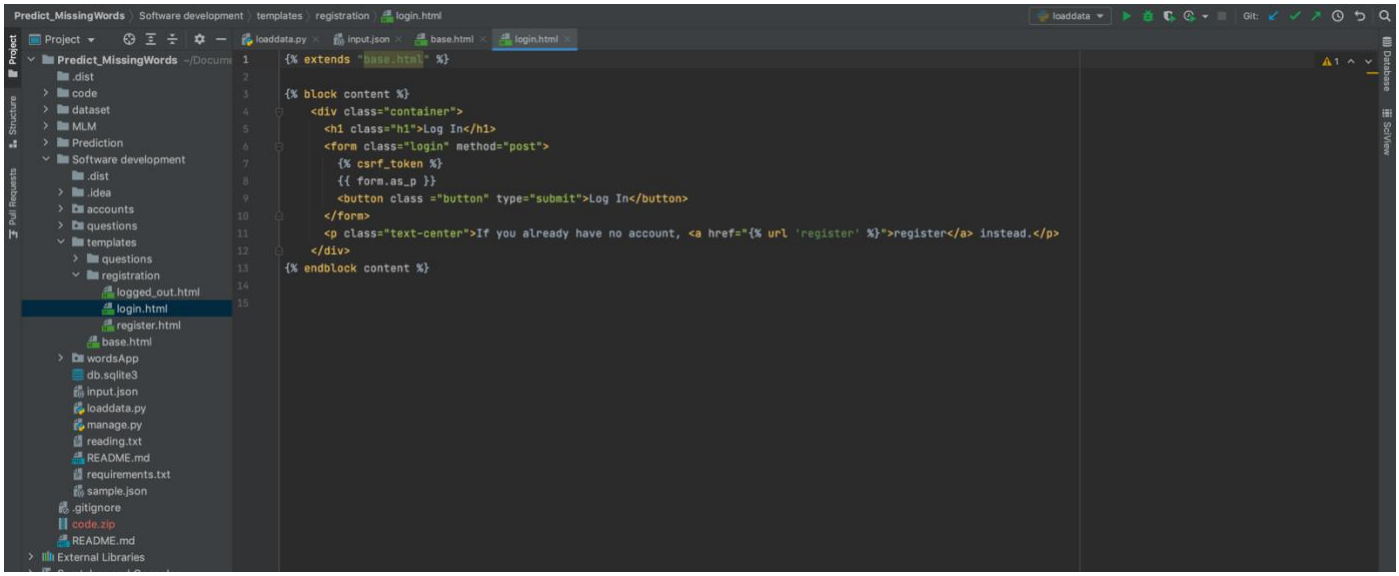
```
        sentences = Sentence.objects.all().order_by(
            '-created_at'
        )
    sentences_ctx = []

    for sent in sentences:
        current_pos = 0
        blank_data = []
        for blank in sent.blanks.all():
            sub = Submission.objects.filter(
                user=request.user,
                blank_=_blank
            ).first()

            blank_data.append({
                'id': blank.id,
                'half': sent.text[current_pos:blank.word_position],
                'choices': [{
                    'id': word.id,
                    'word': word.text
                } for word in blank.words.all()],
                'answer': sub,
                'is_correct': sub.word.text == blank.correct_answer if sub else None
            })
            current_pos = blank.word_position

        sentences_ctx.append({
            "id"_: sent.id,
            "blanks":blank_data,
            "last_half":sent.text[current_pos:]
        })
    return render(
        request, 'questions/index.html', {
            'sentences': sentences_ctx,
            'level': level
        }
    )
```

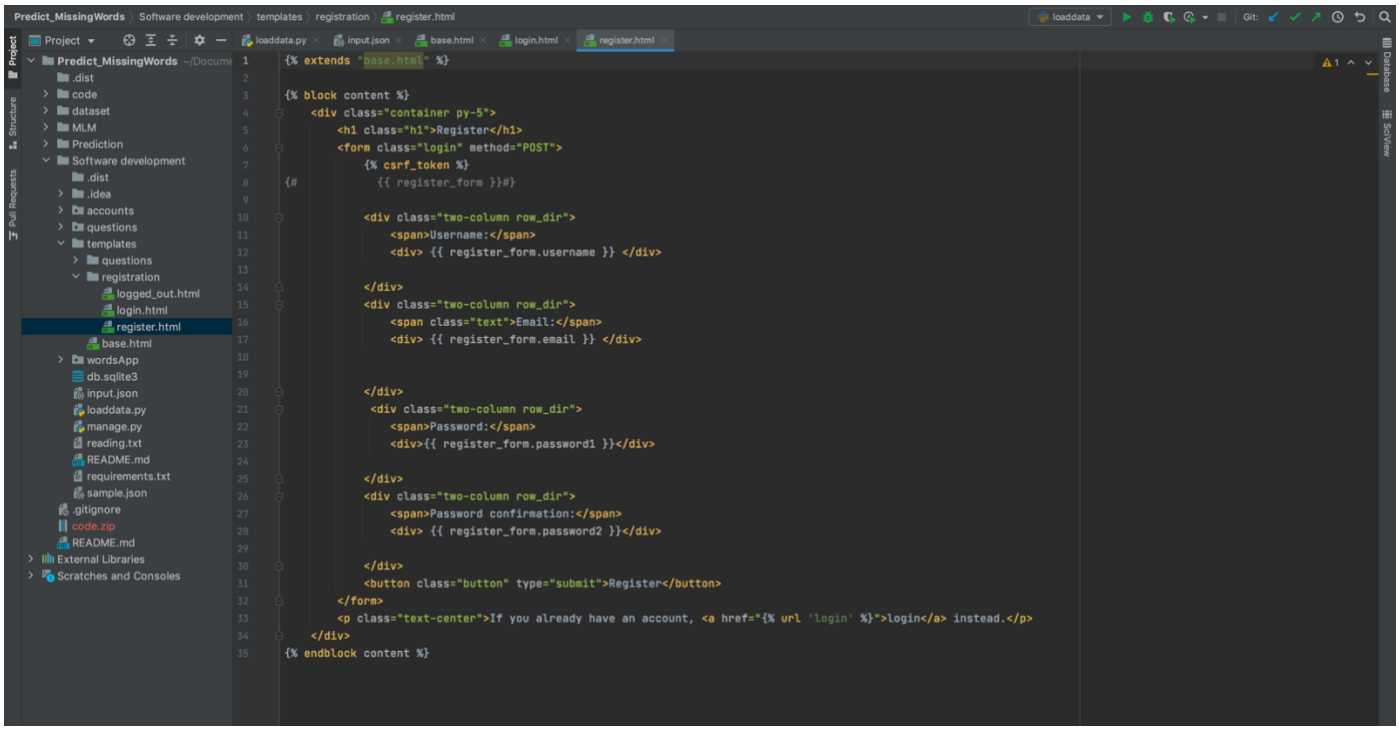
پوشه Template:

در template فایل های مربوط به html پروژه اینجا تعریف میشوند که یک صفحه‌ی ابتدایی داریم که لینک css و javascript را داده ایم و باقی صفحات می‌توانند از آن extend شوند.



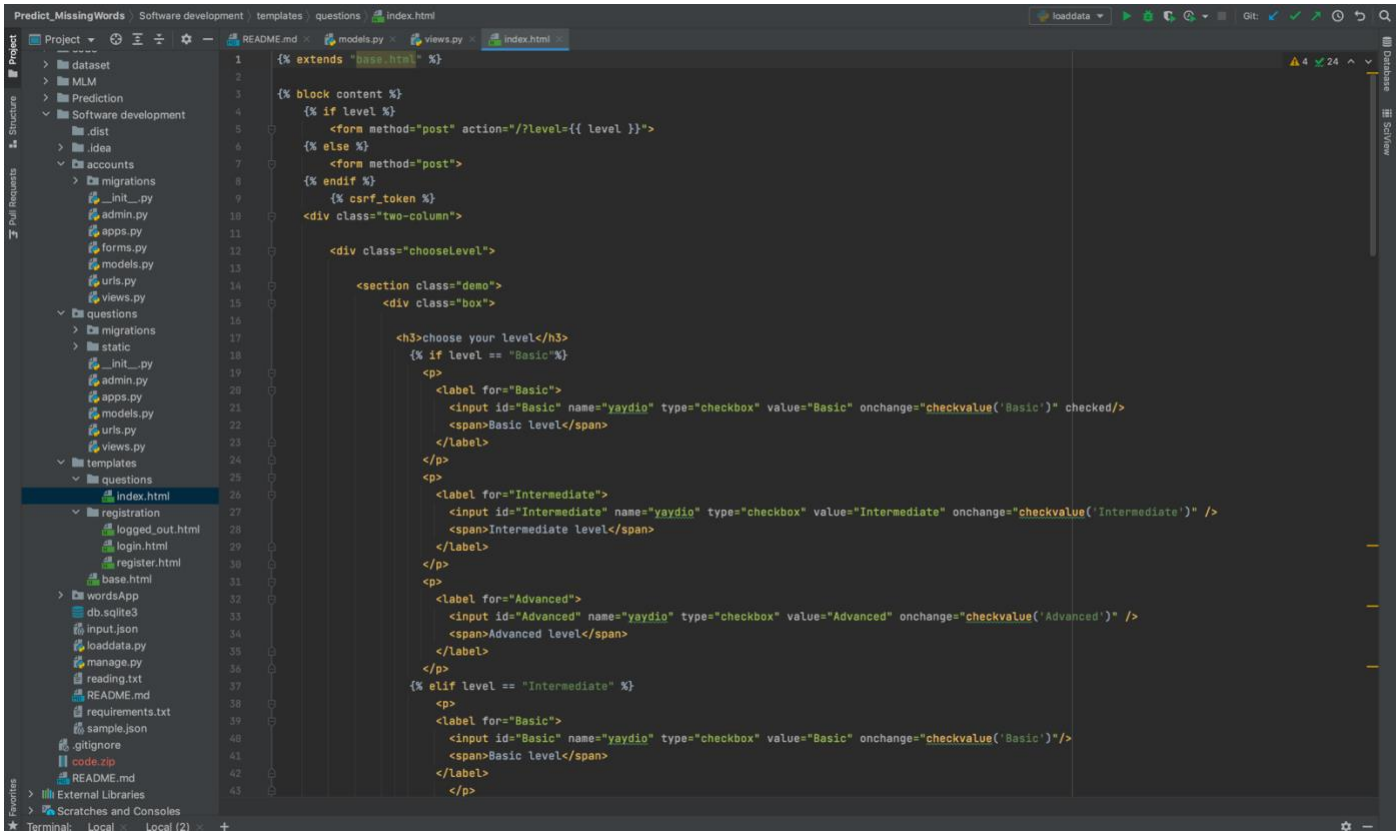
صفحه‌ی ورود و ثبت نام:



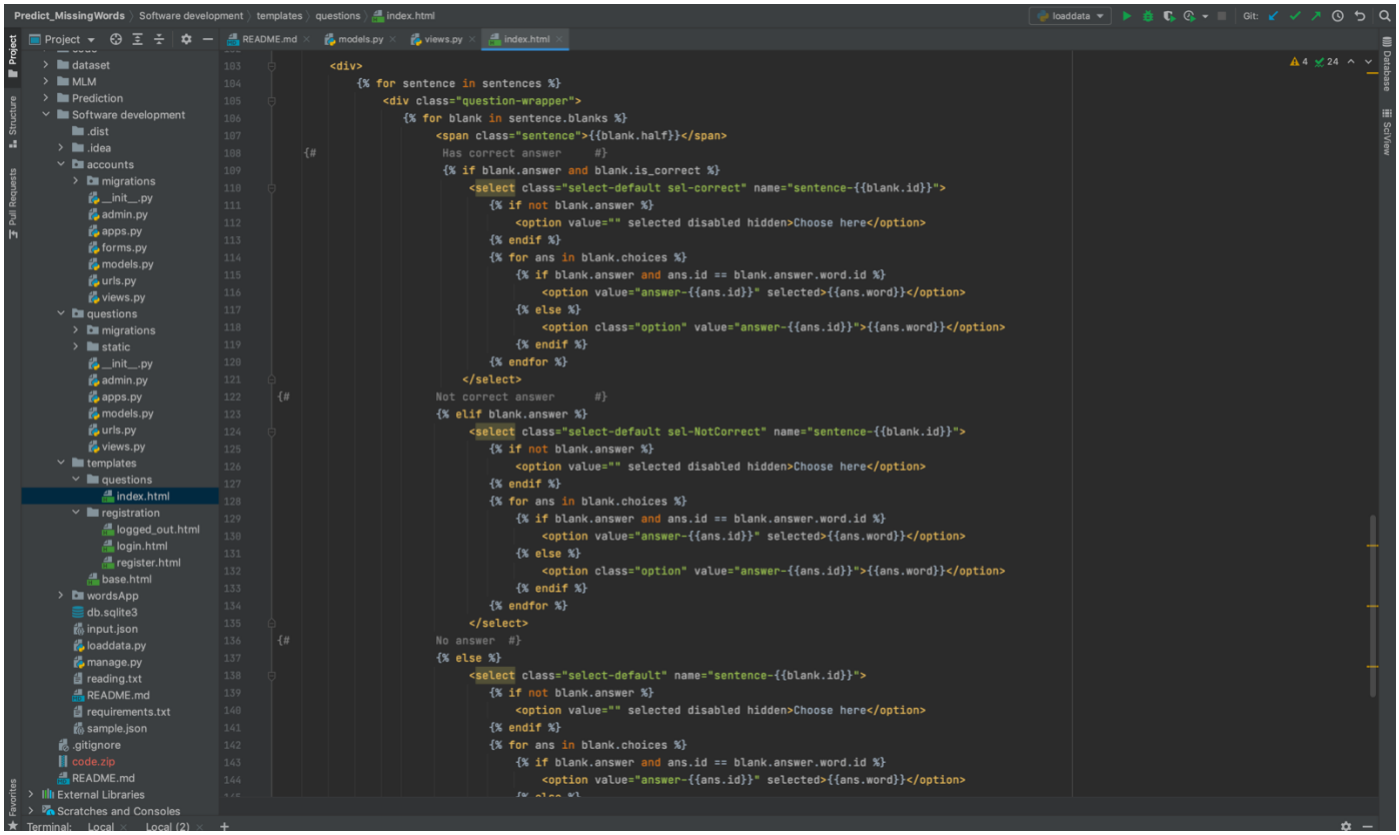


صفحه‌ی جملات:

در ابتدا چک می شود که کاربر سطحی را انتخاب کرده است یا خیر. متناسب با سطحی که کاربر انتخاب می‌کند جملات نشان داده می‌شوند.

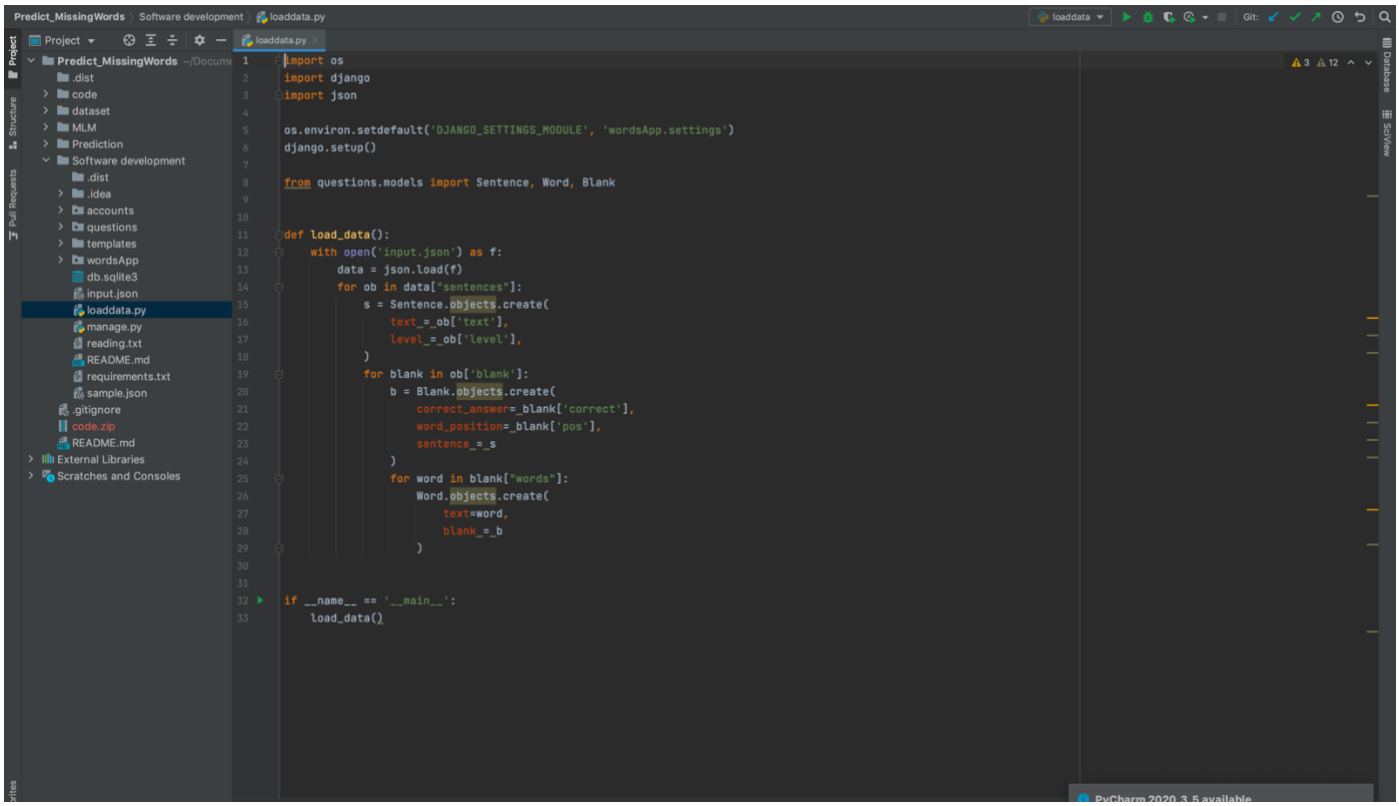


روی تمام جملات در دیتابیس پیمایش صورت میگیرد و روی هر جمله، تمام جای خالی های موجود به کاربر نشان داده می‌شوند و متناسب با اینکه کاربر چه کلمه ای را برای هر جای‌خالی انتخاب کند، اگر کلمه‌ی صحیح باشد رنگ آن سبز می شود و اگر غلط باشد رنگ آن قرمز می‌شود.



فایل Loaddata:

در این فایل، input.json را میخوانیم و باتوجه به مدل دیتابیسی که طراحی کرده‌ایم جملات، جای‌خالی‌ها و کلمات ذخیره می‌شوند.



برای دیدن کد پروژه میتوانید به آدرس زیر مراجعه کنید:

https://github.com/mohadesehjm/Predict_MissingWords/

منابع:

http://jalammar.github.io/illustrated-transformer/

https://medium.com/analytics-vidhya/fine-tuning-bert-language-model-to-get-better-results-on-text-classification-3dac5e3c348e

https://towardsdatascience.com/masked-language-modelling-with-bert-7d49793e5d2c

https://www.tensorflow.org/text/tutorials/fine_tune_bert

https://github.com/Shivampanwar/Bert-text-classification