به نام خدا

پردازش زبان‌های طبیعی

گزارش تمرین سوم

ترک تشخیص و تصحیح غلط‌های املایی متن

سید پوریا لقایی ۹۹۲۱۰۶۹۱

مجتبی زمانی ایمنی ۹۹۲۱۰۶۶۷

مهدی کافی ۹۹۲۱۰۷۵۳

فهرست مطالب

مقدمه

در این تمرین سعی کردیم با استفاده از transformer، مدل n-gram و همچنین Edit distance غلط‌های املایی متن را تشخیص دهیم و آن‌ها را اصلاح کنیم. برای این‌کار از ParsBERT که یک مدل مبتنی بر transformer برای درک زبان فارسی می‌باشد، مدل tri-gram که بر روی دیتاست اخبار می‌باشد و همچنین از توابع مربوط به Edit distance برای محاسبه فاصله کلمات استفاده شده است.

خواندن داده‌ها از گوگل درایو

در این تمرین از دیتاست اخبار که در 4 حوزه سیاست، اقتصاد، ورزش و فرهنگ می‌باشد، استفاده شده است تا در مدل n-gram از آن استفاده کنیم. ابتدا این داده‌ها که در 4 فایل زیپ قرار دارند از درایو خوانده شده و سپس unzip شده است. برای محاسبه احتمالات توسط مدل n-gram به این داده‌ها نیاز داریم تا بتوانیم کلمات بعدی را پیش‌بینی کنیم و با استفاده از آن و تلفیق کردن با دیگر مدل‌ها و توابع، غلط‌های املایی را تصحیح کنیم.

```
# download each subject from google drive
! gdown --id 1dd7-hzTbhffR2mqRPok9wbtAbWWfoUqW
! unzip economics.zip

! gdown --id 1YRH6TL6uPTSweszOOcPUT2tvzVbI_zJS
! unzip politics.zip

! gdown --id 1L0OoEOy2XJUgIDbnoWh-l8bhK31kOE9I
! unzip sports.zip

! gdown --id 1LskvRA05fTXcTMgWsvfd302C8nG2Eq_o
! unzip cultural.zip
```

clone از گیت‌هاب

برای اینکه از داده‌های نوت بوک تدریس‌شده در کلاس استفاده کنیم، آن را در نوت‌بوک خود clone کرده و سپس به همان مسیری که در حال اجرا هستیم منتقل می‌کنیم.

```
!git clone https://github.com/language-ml/2-nlp-language-modeling.git

mv 2-nlp-language-modeling/* ./
```

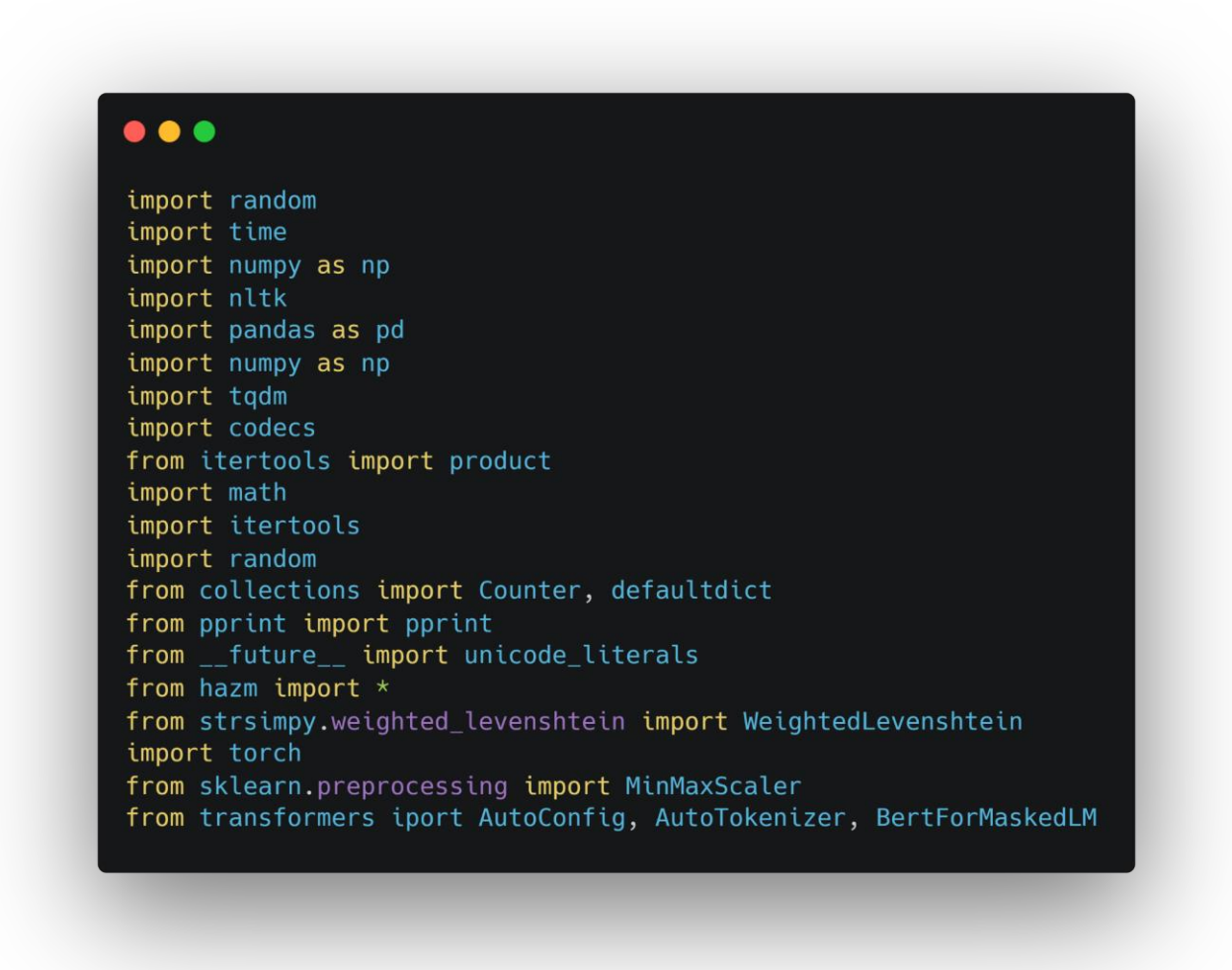
نصب پکیج‌های لازم

در این پروژه نیاز به تعدادی پکیج داریم که باید نصب شوند. از آن‌جایی که روی متن فارسی در حال پردازش هستیم نیاز به ابزار hazm می‌باشد تا با استفاده از آن کارهای پیش‌پردازش را انجام دهیم. چون از transformer استفاده می‌کنیم باید پکیج آن نصب شود. برای محاسبه edit distance ها از Weighted Levenshtein Distance استفاده می‌شود که باید پکیج strsimpy نصب شود.

```
!pip install hazm
!pip install transformers
!pip install strsimpy
```

Import کردن پکیج‌های لازم

حال پکیج‌هایی که لازم است را در نوت بوک import می‌کنیم که یک سری از آن‌ها مربوط به پکیج‌های نصب شده و مابقی مربوط به پکیج‌های از قبل موجود می‌باشد.



```
import random
import time
import numpy as np
import nltk
import pandas as pd
import numpy as np
import tqdm
import codecs
from itertools import product
import math
import itertools
import random
from collections import Counter, defaultdict
from pprint import pprint
from __future__ import unicode_literals
from hazm import *
from strsimpy.weighted_levenshtein import WeightedLevenshtein
import torch
from sklearn.preprocessing import MinMaxScaler
from transformers iport AutoConfig, AutoTokenizer, BertForMaskedLM
```

خواندن از فایل و نمونه‌برداری از آن

همان‌طور که ذکر شد دیتاست اخبار متشکل از 4 حوزه بوده که به نوت بوک اضافه شده است. برای خواندن از این فایل‌ها و نمونه‌برداری از آن‌ها یک تابع تعریف شده است که یک فایل را خوانده و به تعدادی که به عنوان پارامتر می‌گیرد، خبر برمی‌دارد.

```python
def read_file_and_sample(file, sample_size):
  file_lines = []
  with open(file) as f:
    for line in f:
      file_lines.append(line)
    sampled_file = random.sample(file_lines, sample_size)
  return sampled_file
```

حال با استفاده از این تابع از هر حوزه۳۰۰/۰۰۰ خبر برداشته و سپس آن‌ها را با هم ادغام می‌کنیم. دلیل اینکه چرا از تمام دیتاست‌های موجود برای ساخت مدل زبانی ngram استفاده نمی‌شود این است که به میزان رم بیشتری (حتی بیشتر از ۲۵ گیگابایت رم موجود در colab) نیاز داریم که برای ما در دسترس نبود.

```python
# sample 300,000 news from each document
sampled_politics =
read_file_and_sample('politics.txt', 300000)
sampled_economics =
read_file_and_sample('economics.txt',300000)
sampled_sports =
read_file_and_sample('sports.txt',300000)
sampled_cultural =
read_file_and_sample('cultural.txt',300000)

# combining all samples
all_samples = sampled_politics + sampled_economics +
sampled_sports + sampled_cultural
```

پیش‌پردازش

پس از خواندن اخبار موجود در دیتاست‌ها، فرآیند پیش‌پردازش را بر روی آن‌ها اجرا می‌کنیم تا برای آموزش مدل trigram مناسب شوند. در ابتدا هر خبر نرمال می‌شود تا شکل ظاهری کلمات به فرم استاندارد تبدیل شوند. سپس جملات هر خبر جداسازی می‌شوند. سپس به لیستی نیاز داریم که تمامی جملات به صورت رشته در آن باشند به این منظور از chain استفاده کردیم و تمام جملات را در لیست sentences ذخیره کردیم. در نهایت لیستی از stopwordهای زبان فارسی را در لیست stopwords ذخیره کردیم.

```
normalizer = Normalizer()
all_samples_normalized = [normalizer.normalize(x) for x in tqdm.tqdm_notebook(all_samples)]
all_sentences = [sent_tokenize(x) for x in tqdm.tqdm_notebook(all_samples_normalized)]
sentences = list(itertools.chain(*all_sentences))
stopwords = [normalizer.normalize(x.strip()) for x in codecs.open('farsi/stopwords.txt','r','utf-
8').readlines()]
```

مدل n-gram

در این بخش از قطعه کدی که دکتر عسگری در کلاس برای مدل زبانی n-gram توضیح دادن کمک گرفته است و تابعی به آن اضافه شده است با عنوان sent_probability که این تابع n-gramای را دریافت می‌کند و احتمال آن n-gram را براساس مدل آموزش دیده، برمی‌گرداند. این تابع در ابتدا رشته‌ای که متشکل از تعداد n کلمه است را به لیستی از n کلمه تبدیل می‌کند. سپس کلماتی که کمتری مساوی یک بار در کل دیتاست دیده شده‌اند را با <UNK> جایگزین می‌کند. سپس n-gram را به n-gramای تبدیل می‌کند که برای مدل زبانی قابل شناسایی باشد و در نهایت احتمال n-gram را در خروجی برمی‌گرداند.

```
def sent_probability(self, test_data):
    test_tokens = test_data.split()
    test_tokens = self.replace_singletons(test_tokens)
    known_ngram = self._convert_oov(tuple(test_tokens))
    prob = self.model[known_ngram]
    return prob
```

تعریف trigram

در این تمرین از مدل trigram استفاده کردیم که باعث می‌شود احتمال تمام سه‌تایی‌های موجود در دیتاست را محاسبه کند و در زمان نیاز این احتمال را به ما بدهد.

```
#3_gram module
language_model = LanguageModel(sentences, 3)
```

مدل ParsBERT

در این تمرین برای آموزش مدل از یک transformer از قبل train شده با نام ParsBERT استفاده کردیم که اطلاعات آن در https://huggingface.co/HooshvareLab/bert-base-parsbert-uncased موجود است. در این بخش تمام ۴۲۰۰۰ کلمه موجود در ترنسفورمر را در لیست vocabulary ذخیره کردیم و تعداد زیادی از این لغات دارای عبارت «##» بودند که آنها را در حلقه حذف کردیم.

```
#load model v3.0
model_name_or_path = "HooshvareLab/bert-fa-zwnj-base"
config = AutoConfig.from_pretrained(model_name_or_path)
tokenizer = AutoTokenizer.from_pretrained(model_name_or_path)
vocabulary = [tokenizer.decode(idx) for idx in range(42000)]
for j, voc in enumerate(vocabulary):
    if "##" in voc:
        vocabulary[j] = voc[2:]
model = BertForMaskedLM.from_pretrained(model_name_or_path)
```

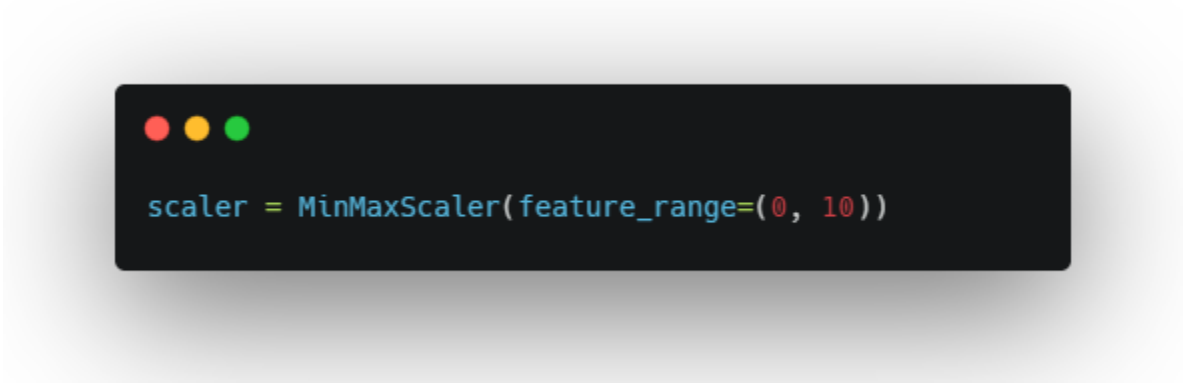
محاسبه Edit Distance

برای محاسبه edit distance ها از Weighted Levenshtein Distance استفاده شد بدین صورت که پنالتی مورد نظر برای insertion و deletion برابر 1.5 و برای substitution برابر 1 در نظر گرفته شد زیرا تشخیص دادیم که substitution باید پنالتی کمتری داشته باشد تا کلمات به درستی اصلاح شوند.

```
def insertion_cost(char):
    return 1.5

def deletion_cost(char):
    return 1.5

def substitution_cost(char_a, char_b):
    return 1.0

weighted_levenshtein = WeightedLevenshtein(
    substitution_cost_fn=substitution_cost,
    insertion_cost_fn=insertion_cost,
    deletion_cost_fn=deletion_cost)
```

MinMaxScaler

هنگام محاسبه امتیازات مدل‌های مختلف اعدادی که بدست می‌آید بزرگ هستند و برای اینکه بتوانیم آن‌ها را مقایسه کنیم با استفاده از MinMaxScaler همه مقادیر را بین 0 تا 10 می‌آوریم.



```
scaler = MinMaxScaler(feature_range=(0, 10))
```

تلفیق امتیازات

هر یک از مدل‌های تعریف شده و همچنین Edit Distance بدست آمده دارای یک امتیاز می‌باشد. با استفاده از یک تابع هر امتیاز را در ضریب مورد نظر ضرب کرده و سپس با جمع آن‌ها امتیاز کلی بدست می‌آید. امتیاز بخش edit distance از بقیه بخش ها بیشتر است. دلیل آن هم این است که transformer برای جایگذاری کلمه MASK شده کاری به نزدیک بودن دو کلمه با هم ندارد (معمولا در صورت رخداد غلط املایی فاصله بین لغات در حد دو الی سه تغییر است). حال ما وزن بخش edit distance را زیاد می کنیم تا این بخش مدیریت کار را تا حدودی در دست داشته باشد. البته باز نیز مقدار آن باید با توجه به مقادیر دیگر تنظیم شود و اینگونه نشود که تاثیر transformer و مدل n-gram را به طور کلی از بین ببرد.

```
def combine_scores(parsbert_scores, meds,
n_gram_scores, alpha=1, beta=5, gamma=1):
    return alpha*parsbert_scores - beta*meds +
gamma*n_gram_scores
```

تلفیق مدل‌ها و محاسبه امتیاز

در این بخش تابعی داریم که جمله، کلمه‌ای که باید برای تصحیح غلط املایی بررسی شود و جایگاه آن کلمه در جمله را می‌گیرد و در ابتدا با استفاده از parsbert سعی می‌کند کلمات پیشنهادی به جای آن کلمه را برگرداند و در ابتدا ۵۰۰ کلمه برتر از نظر ترنسفورمر، به این صورت بررسی می‌شوند که در صورتیکه فاصله ویرایشی آن تا کلمه اصلی، کمتر از ۲ باشد کلمه پیشنهادی ترنسفورمر را با کلمه اصلی جایگزین می‌کند. در صورتیکه در بین ۵۰۰ کلمه چنین کلمه‌ای با خصوصیات گفته شده یافت نشود سعی می‌کنیم از هر ۳ مدل کمک بگیریم به این صورت که از امتیاز تمام ۴۲۰۰۰ کلمه موجود در ترنسفورمر استفاده می‌کنیم و همچنین فاصله کلمه اصلی را با تمام ۴۲۰۰۰ کلمه موجود در ترنسفورمر محاسبه می‌کنیم و سپس کلمه اصلی و کلمه قبل و بعد آن را در نظر می‌گیریم و به جای کلمه اصلی تمام ۴۲۰۰۰ کلمه ممکن را جایگزین می‌کنیم و احتمال هر سه تایی را بررسی می‌کنیم. سپس امتیاز هر سه مدل را با تابع combine_scores تلفیق می‌کنیم و کلمه‌ای که بیشترین امتیاز را به دست آورد با کلمه اصلی جایگزین می‌کنیم.

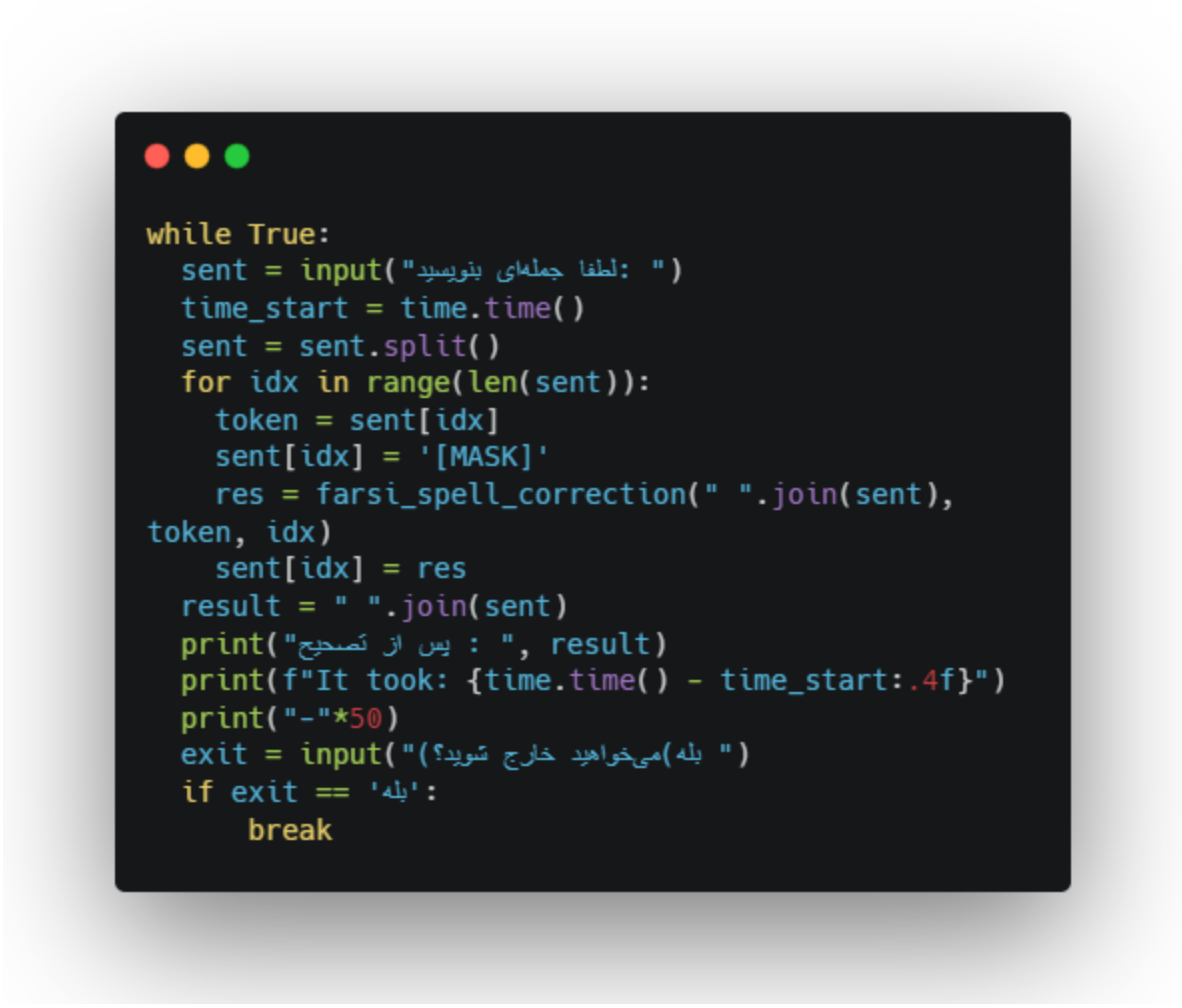
```python
def farsi_spell_correction(text, token, idx):
  inputs = tokenizer(text, return_tensors='pt')
  with torch.no_grad():
    logits = model(**inputs).logits.squeeze()
  mask_token_index = (inputs.input_ids == tokenizer.mask_token_id)[0].nonzero(as_tuple=True)[0]
  predicted_token_id = logits[mask_token_index]
  copy_parsbert_scores = torch.clone(predicted_token_id)
  for _ in range(500):
      i = copy_parsbert_scores.argmax()
      if weighted_levenshtein.distance(tokenizer.decode(i), token) < 2:
          return tokenizer.decode(i)
      copy_parsbert_scores[0, i] = -np.Inf
  del copy_parsbert_scores
  predicted_token_id.resize_(42000, 1)
  scaled_parsbert_scores = scaler.fit_transform(predicted_token_id)
  meds = np.array([weighted_levenshtein.distance(token, vocab) for vocab in vocabulary])
  scaled_meds = scaler.fit_transform(meds.reshape(-1, 1))
  text = '<s> ' + text + ' </s>'
  text = text.split()
  n_gram_scores = [language_model.sent_probability(text[idx-1]+" "+vocab+ " "+text[idx+1]) for vocab in
vocabulary][0]
  n_gram_scores = np.array(n_gram_scores)
  n_gram_socres = scaler.fit_transform(n_gram_scores.reshape(-1, 1))
  combined_scores = combine_scores(scaled_parsbert_scores, scaled_meds, n_gram_scores, 1, 5, 1)
  return tokenizer.decode(combined_scores.argmax())
```

نحوه اجرای برنامه

ابتدا یک حلقه while True قرار می‌دهیم تا برای همیشه ورودی بگیرد (البته یک شرط برای پایان حلقه در بدنه آن تعریف شده است که توضیح داده می‌شود). حال یک جمله به عنوان ورودی گرفته می‌شود. همچنین چون می‌خواهیم زمان اجرای هر کدام از test caseها را محاسبه کنیم، توسط time زمان شروع را ذخیره می‌کنیم. سپس تمام کلمات متن را در یک لیست ذخیره می‌کنیم.

حال در یک حلقه به طول لیست، پیمایش انجام می‌دهیم. بدین صورت که هربار کلمه مربوطه را ذخیره کرده و سپس آن را در لیست MASK می‌کنیم. در مرحله‌ی بعد تابع farsi_spell_correction را فراخوانی می‌کنیم و ورودی‌های آن که join شده لیست، کلمه ذخیره شده و اندیس آن کلمه در لیست است را به تابع می‌دهیم. سپس خروجی تابع که کلمه تصحیح شده است را جایگزین آن اندیس در لیست می‌کنیم.

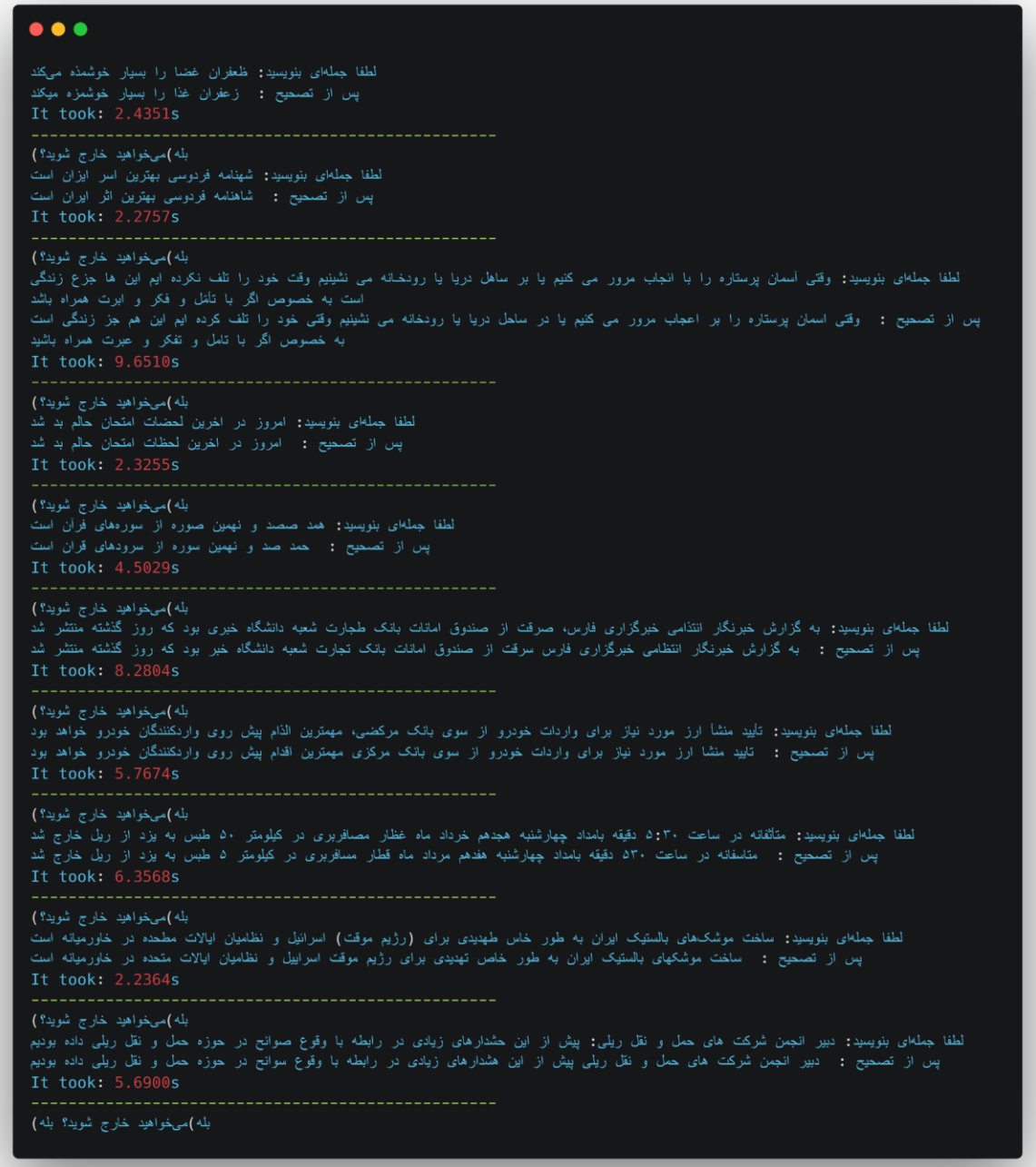
سپس تمام عناصر لیست را با هم join کرده و حاصل را نمایش می‌دهیم. زمان که این عملیات طول کشید را نیز توسط time فعلی و time شروع ذخیره می‌کنیم. حال از کاربر سوال می‌شود که آیا قصد دارد خارج شود یا خیر. اگر بله را وارد کند حلقه while به اتمام رسیده و برنامه به اتمام می‌رسد و در غیر این صورت حلقه دوباره اجرا می‌شود.



```
while True:
  sent = input("لطفا جمله‌ای بنویسید: ")
  time_start = time.time()
  sent = sent.split()
  for idx in range(len(sent)):
    token = sent[idx]
    sent[idx] = '[MASK]'
    res = farsi_spell_correction(" ".join(sent),
token, idx)
    sent[idx] = res
  result = " ".join(sent)
  print("پس از تصحیح : ", result)
  print(f"It took: {time.time() - time_start:.4f}")
  print("-"*50)
  exit = input("(بله ")می‌خواهید خارج شوید؟
  if exit == 'بله':
      break
```

آزمایش بر روی تعدادی جمله

در نهایت تعدادی جمله که غلط‌های املایی در آنها وجود داشتند را به مدل دادیم و جمله اولیه و نتیجه را در زیر آورده‌ایم.


لطفا جمله‌ای بنویسید: طعفران غضا را بسیار خوشمذه می‌کند
پس از تصحیح : زعفران غذا را بسیار خوشمزه میکند
It took: 2.4351s
بله)می‌خواهید خارج شوید؟(
لطفا جمله‌ای بنویسید: شهنامه فردوسی بهترین اسر ایزان است
پس از تصحیح : شاهنامه فردوسی بهترین اثر ایران است
It took: 2.2757s
بله)می‌خواهید خارج شوید؟(
لطفا جمله‌ای بنویسید: وقتی آسمان پرستاره را با انجاب مرور می کنیم یا بر ساهل دریا یا رودخانه می نشینیم وقت خود را تلف نکرده ایم این ها جزع زندگی است به خصوص اگر با تامل و فکر و ابرت همراه باشد
پس از تصحیح : وقتی اسمان پرستاره را بر اعجاب مرور می کنیم یا در ساحل دریا یا رودخانه می نشینیم وقتی خود را تلف کرده ایم این هم جز زندگی است به خصوص اگر با تامل و تفکر و عبرت همراه باشید
It took: 9.6510s
بله)می‌خواهید خارج شوید؟(
لطفا جمله‌ای بنویسید: امروز در اخرین لحضات امتحان حالم بد شد
پس از تصحیح : امروز در اخرین لحظات امتحان حالم بد شد
It took: 2.3255s
بله)می‌خواهید خارج شوید؟(
لطفا جمله‌ای بنویسید: همد صمصد و نهمین صوره از سوره‌های فران است
پس از تصحیح : حمد صد و نهمین سوره از سرودهای قران است
It took: 4.5029s
بله)می‌خواهید خارج شوید؟(
لطفا جمله‌ای بنویسید: به گزارش خبرنگار انتذامی خبرگزاری فارس، صرقت از صندوق امانات بانک طجارت شعبه دانشگاه خبری بود که روز گذشته منتشر شد
پس از تصحیح : به گزارش خبرنگار انتظامی خبرگزاری فارس سرقت از صندوق امانات بانک تجارت شعبه دانشگاه خبر بود که روز گذشته منتشر شد
It took: 8.2804s
بله)می‌خواهید خارج شوید؟(
لطفا جمله‌ای بنویسید: تایید منشأ ارز مورد نیاز برای واردات خودرو از سوی بانک مرکضی، مهمترین الذام پیش روی واردکنندگان خودرو خواهد بود
پس از تصحیح : تایید منشا ارز مورد نیاز برای واردات خودرو از سوی بانک مرکزی مهمترین اقدام پیش روی واردکنندگان خودرو خواهد بود
It took: 5.7674s
بله)می‌خواهید خارج شوید؟(
لطفا جمله‌ای بنویسید: متأثفانه در ساعت ۵:۳۰ دقیقه بامداد چهارشنبه هجدهم خرداد ماه غطار مصافربری در کیلومتر ۵۰ طبس به یزد از ریل خارج شد
پس از تصحیح : متاسفانه در ساعت ۵۳۰ دقیقه بامداد چهارشنبه هفدهم مرداد ماه قطار مسافربری در کیلومتر ۵ طبس به یزد از ریل خارج شد
It took: 6.3568s
بله)می‌خواهید خارج شوید؟(
لطفا جمله‌ای بنویسید: ساخت موشک‌های بالستیک ایران به طور خاس طهدیدی برای (رژیم موقت) اسرائیل و نظامیان ایالات مطحده در خاورمیانه است
پس از تصحیح : ساخت موشکهای بالستیک ایران به طور خاص تهدیدی برای رژیم موقت اسراییل و نظامیان ایالات متحده در خاورمیانه است
It took: 2.2364s
بله)می‌خواهید خارج شوید؟(
لطفا جمله‌ای بنویسید: دبیر انجمن شرکت های حمل و نقل ریلی: پیش از این حشدارهای زیادی در رابطه با وقوع صوانح در حوزه حمل و نقل ریلی داده بودیم
پس از تصحیح : دبیر انجمن شرکت های حمل و نقل ریلی پیش از این هشدارهای زیادی در رابطه با وقوع سوانح در حوزه حمل و نقل ریلی داده بودیم
It took: 5.6900s
بله)می‌خواهید خارج شوید؟ بله(