

**درس پردازش زبان طبیعی**

**استاد ممتازی**

**نیما پری فرد**

**402131017**

فهرست تمرین اول پردازش زبان طبیعی

[توجه 2](#_Toc167889631)

[بخش 1 3](#_Toc167889632)

[مشاهده داده و اعمال پیش پردازش های لازم 3](#_Toc167889633)

[بخش 2 4](#_Toc167889634)

[بخش الف 4](#_Toc167889635)

[بخش ب 11](#_Toc167889636)

[بخش ج 13](#_Toc167889637)

[بخش د 17](#_Toc167889638)

[بخش 3 20](#_Toc167889639)

[بخش الف 20](#_Toc167889640)

[بخش ب 22](#_Toc167889641)

[بخش ج 25](#_Toc167889642)

[بخش د 28](#_Toc167889643)

[بخش 4 32](#_Toc167889644)

[سطح token 32](#_Toc167889645)

[سطح entity 33](#_Toc167889646)

# توجه

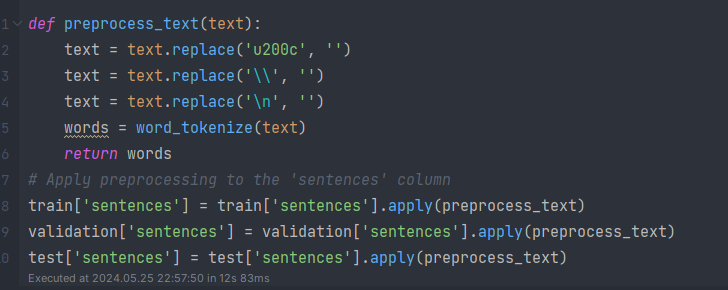
شاید شکل ظاهری بعضی screenshot ها گرفته­شده متفاوت باشد، دلیلش این این است بعضی ها از colab برداشتم و بعضی ها را از pycharm برداشتم.

# بخش 1

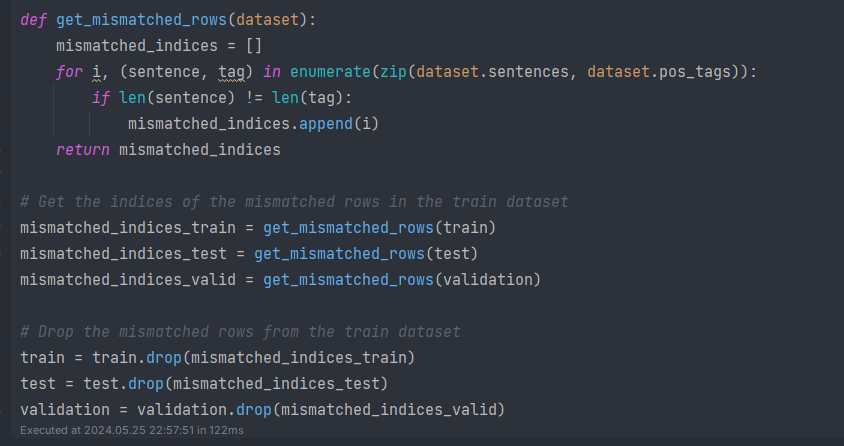
## مشاهده داده و اعمال پیش پردازش های لازم

پیش­پردازش هایی که انجام دادم:

* نیم فاصله ها را حذف کردم چون نمی شناخت آن ها را و بجاش u200c\ می­گذاشت و کلمه جدید به اشتباه تولید می­کرد.
* token بندی کردم

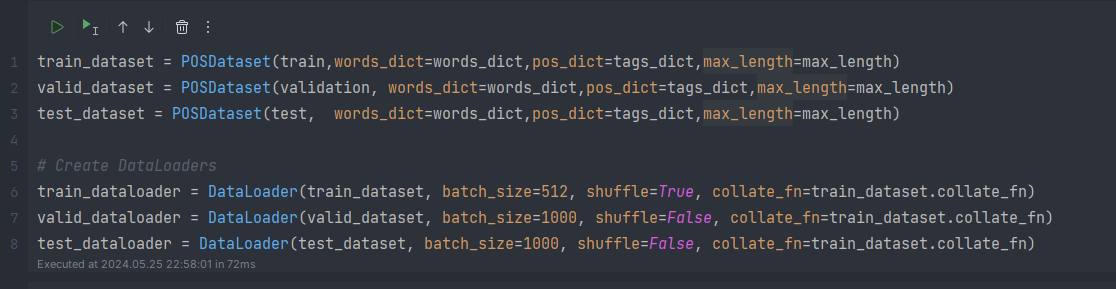


* بعضی داده ها طول جمله شان با طول تگ ها یکسان نبود آن ها را حذف کردم.



* دیتاست مخصوص داده ها به صورت ارث­بری از تابع عمومی دیتاست در پایتورچ درست کردم در آن پدینگ انجام می­شود تا طول جملات یکسان شوند. همچنین وقتی می خواد خروجی بده اندیکس لغات و تگ ها را به جای خود لغات و تگ ها قرار می­دهد.



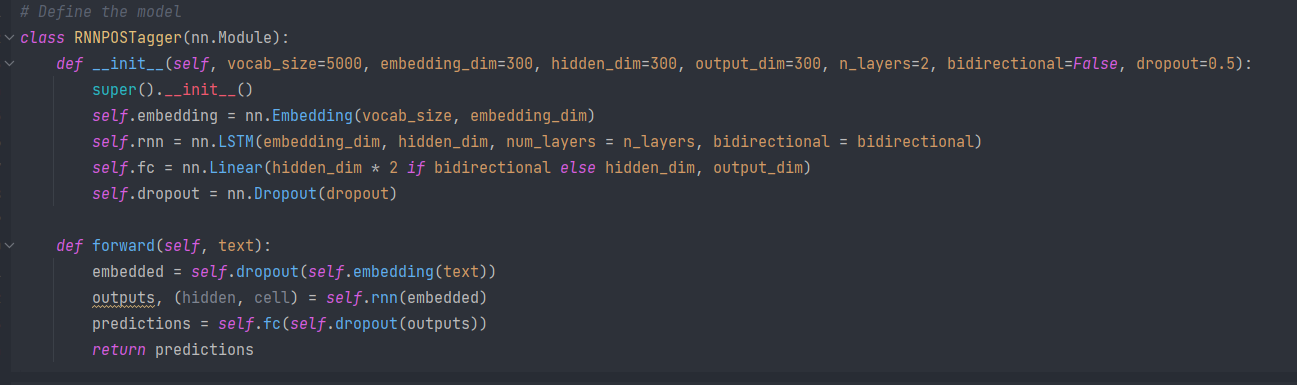


* داده ها بسیار کثیف است چون داده های توییت هست برای همین توقع درصد بالا نباید داشت.

# بخش 2

## بخش الف

کد مدل استفاده شده برای ران کردن قسمت بعدی:



در مدل بالا یک جمله به آن می­دهیم و در نهایت تگ های آن را خروجی می­دهد. البته این قبل آموزش است یادم رفت خروجی بگیرم و بعد در صورت سوال متوجه شدم که می­خواد اینو خروجی بگیرم با وزن های رندم یک مدل ساختم فقط خروجی گرفتم.



معماری شبکه را با استفاده کد زیر نمایش دادم:

A computer screen shot of a program

Description automatically generated

A screenshot of a computer

Description automatically generated

با استفاده از کلاس ModelOptimization مدل را آموزش دادم. تقریبا براین این کلاس از کلاس برای آموزش مدل­ها استفاده کردم.

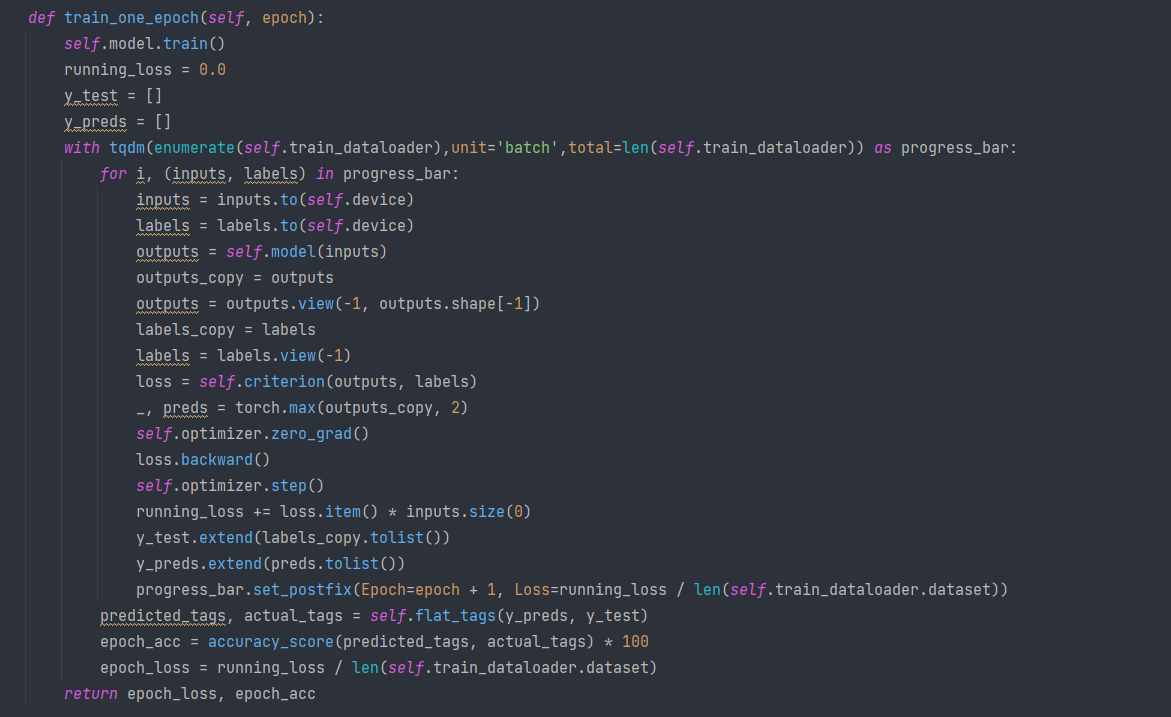
درون وقتی مدل را load می­کنم یک عمل مقداردهی اولیه با استفاده از xavier\_normal مقدار دهی کردم.

A screen shot of a computer code

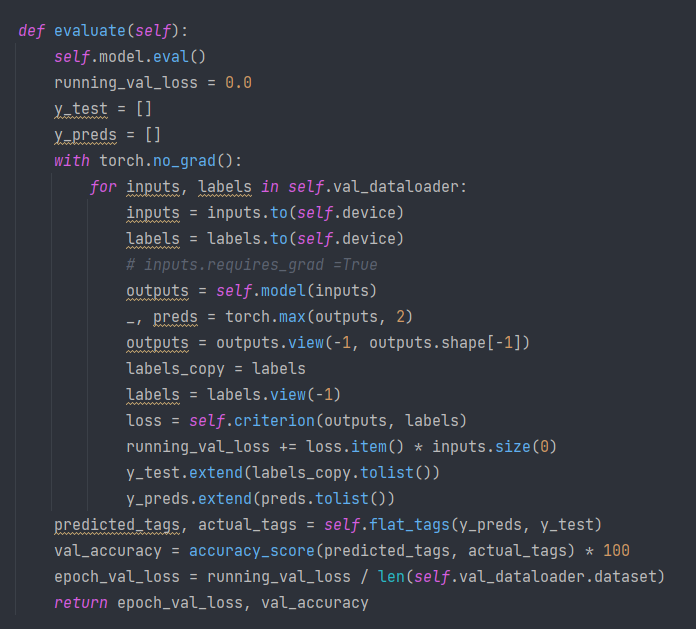
Description automatically generated

A black background with orange and white text

Description automatically generated

با استفاده از تابع مدل را یک epoch آموزش می­دادم. 

با استفاده از تابع زیر مدل را به حالت eval می­بردم تا نتایج روی داده های validation ارزیابی کنم.



با استفاده از تابع زیر خروجی مدل را به حالت flat در می­آورم تا در نهایت بتوانم دقت را خروجی را با لیبل ها مقایسه کنم.

A computer screen shot of text

Description automatically generated

با استفاده از تابع مدل را با تعداد epoch دریافت شده آموزش می­دهم تا نتایج را برای هر epoch خروجی می­دادم.

A computer screen shot of text

Description automatically generated

در نهایت مدل را سیو می­کنم.

A computer screen shot of text

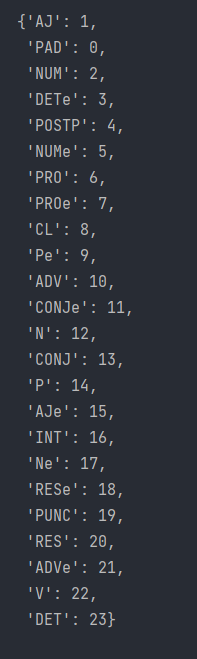
Description automatically generated

برای آموزش مدل که تهش CRF داره، یک کلاس جدید ساختم برای آموزش چون خروجی آن در نهایت از لایهCrf می­اید کمی بعضی از خروجی گرفتن ها متفاوت است.

A screen shot of a computer

Description automatically generated

دیکشنری تگ ها ساختم برای متوجه شدن خروجی ها این دیکشنری قرار دادم.



## بخش ب

با استفاده از داده های validation هایپرپارامترها را tune کردم.

A black background with white lines

Description automatically generated with medium confidence

A screenshot of a computer

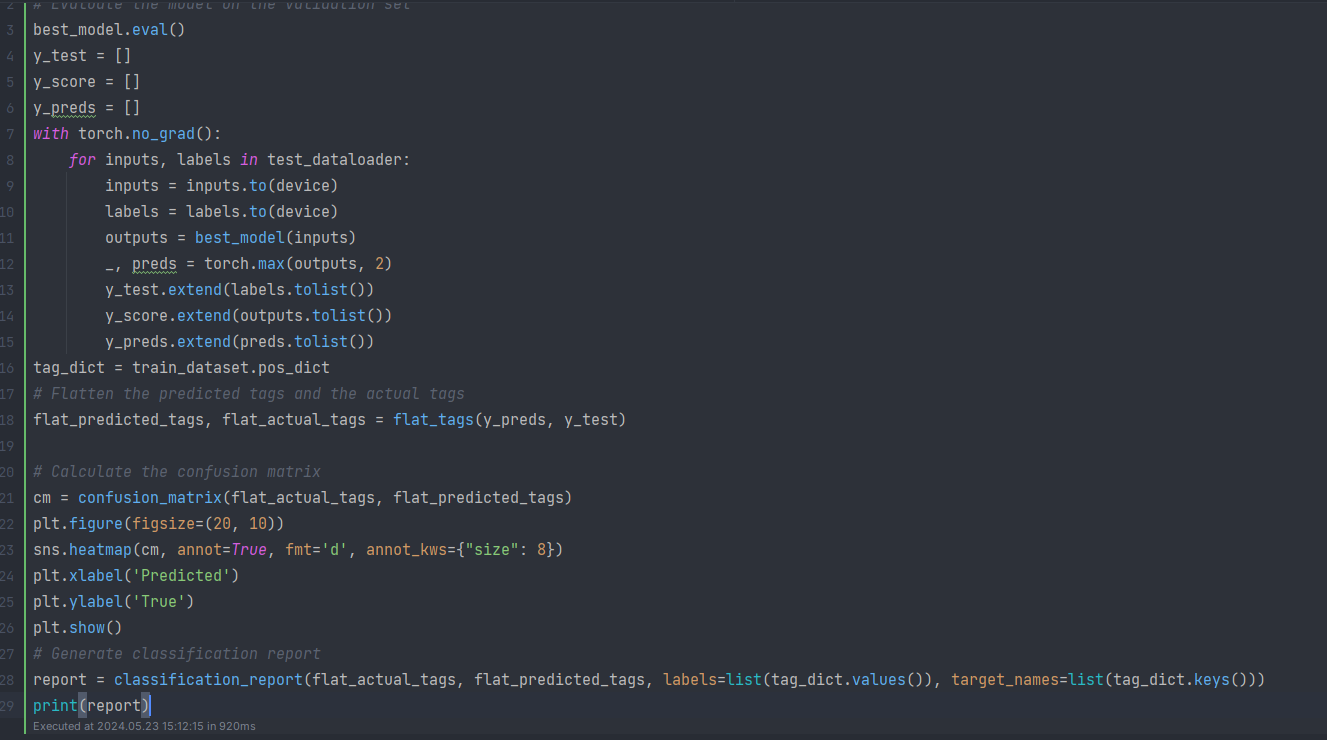
Description automatically generated

اموزش مدل در نهایت:

A screenshot of a computer

Description automatically generated

Precision, recall, f1 score, accuracy برای داده­های تست:



A screenshot of a computer screen

Description automatically generated

## بخش ج

با استفاده از کتابخانه نصب شده یک لایه crf بالای مدل قرار دادم. در نهایت خروجی که دریافت می­کنیم با استفاده از لایه crf که در بالای مدل قرار دادم. برای crf ار کتاب­خانه pytorch-crf استفاده کردم.

A screen shot of a computer code

Description automatically generated

هایپرپارامتر ها را تیون کردم:

A black background with white lines

Description automatically generated

A screenshot of a computer

Description automatically generated

مدل را در نهایت آموزش دادم:

A screenshot of a computer program

Description automatically generated

Precision, recall, f1 score, accuracy برای داده­های تست:

A screenshot of a computer screen

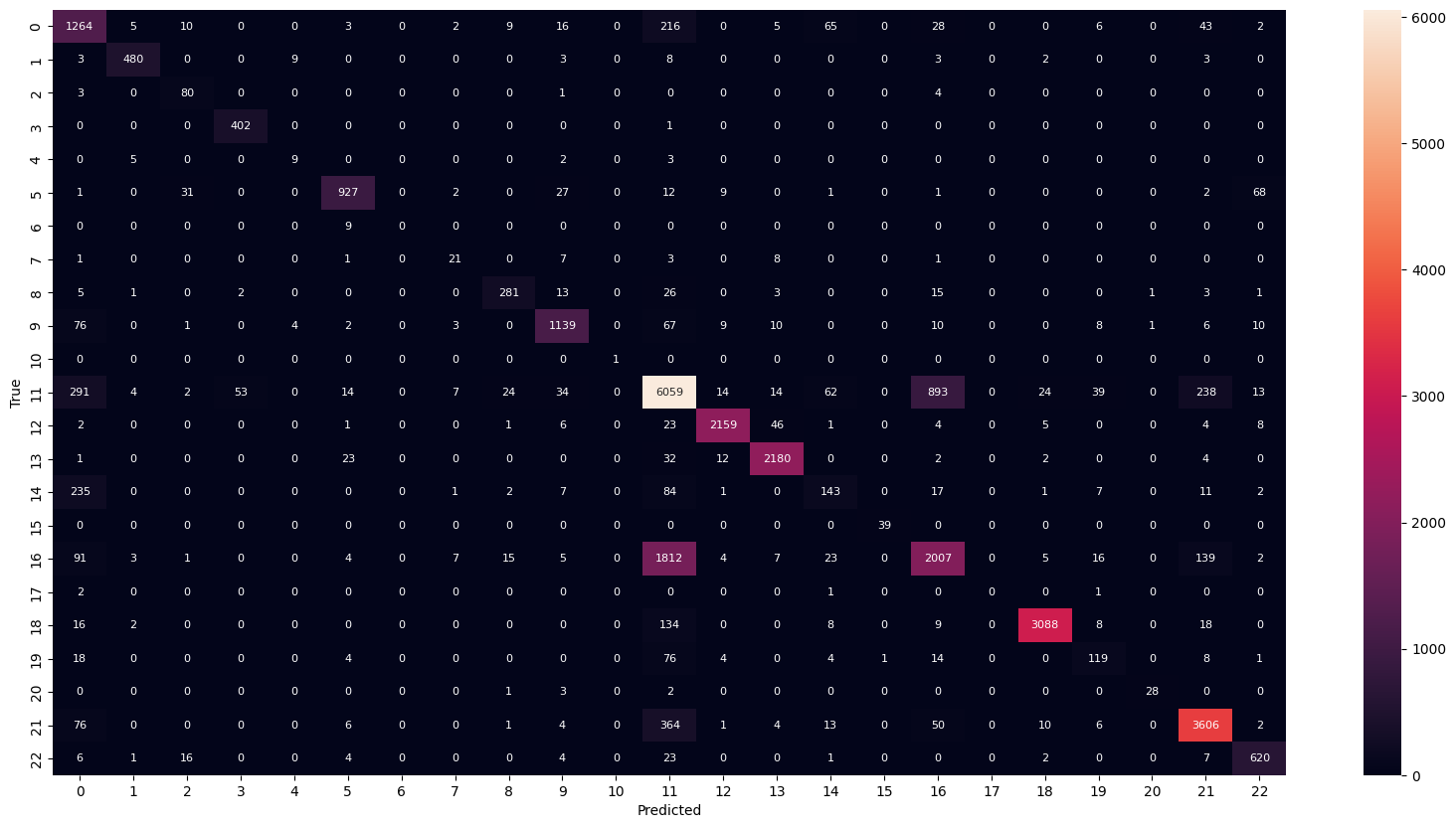
Description automatically generated

## بخش د



از کد بالا برای رسم بهترین ماتریس درهم­ریختگی بهترین مدل های بخش ب و ج رسم می­کند همچنین 5 تا از بدترین نتیجه های هر دو مدل را نمایش می­دهد.

مدل بخش ب:



در لیست پراشتباه ترین موارد در ماتریس درهم­ریختگی برای بهترین مدل بخش ب:

A screenshot of a computer code

Description automatically generated

مدل بخش ج:

A screen shot of a graph

Description automatically generated

در لیست پر اشتباه ترین موارد در ماتریس درهم­ریختگی برای مدل بخش ج:

A screenshot of a computer code

Description automatically generated

انالیز نتایج:

- هر دو مدل تقریبا عملکرد یکسانی از خود نشان دادند، نمی­شه گفت کدوم داره بهتر عمل می­کند چوت تفاوت ها واقعا جزئی است.

- هر دو مدل در برچسب‌های خاصی اشتباهات رایج یکسانی دارند.

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated

- برچسب‌هایی مانند ADJ'و PRO و CON وPUNC و Pو V و DET به طور مداوم توسط هر دو مدل به خوبی تشخیص داده می‌شوند، که نشان می‌دهد این دسته‌ها برای مدل‌ها آسان‌تر هستند.

A screenshot of a computer code

Description automatically generatedA screenshot of a computer code

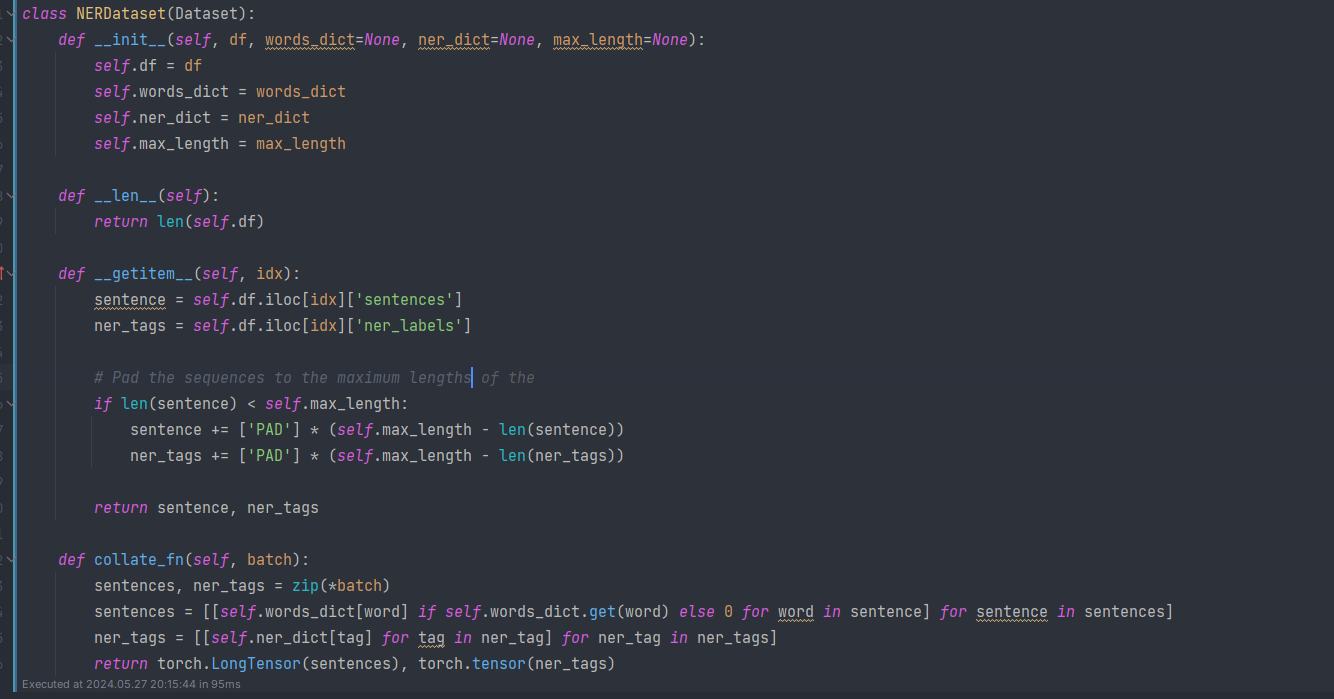
Description automatically generated

- برچسب‌های مثلNe، AJ, N که تعداد بالاتری دارند و خطای آن ها در عملکرد مدل بسیار حساس است نرخ خطای بالاتری دارند. چیزی که مشخص است بیش ترین موردی که مدل با آن مشکل دارد این تگ هاست درصد اشتباه آن به نسبت تعداد آن بالاست یک ایده ای می­توان استفاده کرد، این است در تابع خطا اولویت بیش تری به این کلاس بدهیم این کار را انجام دادم اما نتایج بدتر شد و حدود ده درصد افت نتایج داشتم.

# بخش 3

## بخش الف

کد های این قسمت خیلی جاهایش مثل سوال قبلیست در واقع بیش ترش از قسمت قبل برداشتم فقط اسمش را تغییر دادم برای همین اون کد ها را نمی­ارم برای مثال کد دیتاست که قسمت گزاشتم و با کمک اون پد با داده اضافه می­کردم.

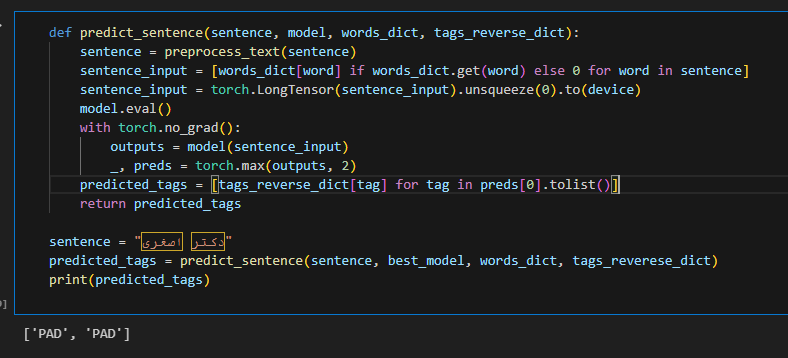


کد مدل استفاده شده برای ران کردن قسمت بعدی:

A screenshot of a computer code

Description automatically generated

در مدل بالا یک جمله به آن می­دهیم و در نهایت تگ های آن را خروجی می­دهد. البته این قبل آموزش است یادم رفت خروجی بگیرم و بعد در صورت سوال متوجه شدم که می­خواد اینو خروجی بگیرم با وزن های رندم یک مدل ساختم فقط خروجی گرفتم.



معماری شبکه را با استفاده کد زیر نمایش دادم:

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

## بخش ب

با استفاده از داده های validation هایپرپارامترها را tune کردم.



A screenshot of a computer program

Description automatically generated

اموزش مدل در نهایت:

A screenshot of a computer program

Description automatically generated

Precision, recall, f1 score, accuracy برای داده­های تست:

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## بخش ج

با استفاده از کتابخانه نصب شده یک لایه crf بالای مدل قرار دادم.

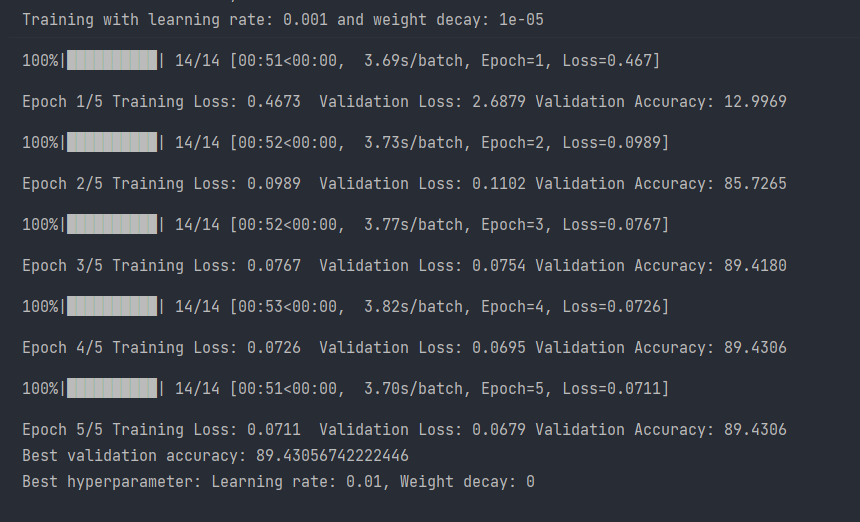
A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

هایپرپارامتر ها را تیون کردم:



مدل را در نهایت آموزش دادم:

A screenshot of a computer program

Description automatically generated

Precision, recall, f1 score, accuracy برای داده­های تست:

A screenshot of a computer screen

Description automatically generated

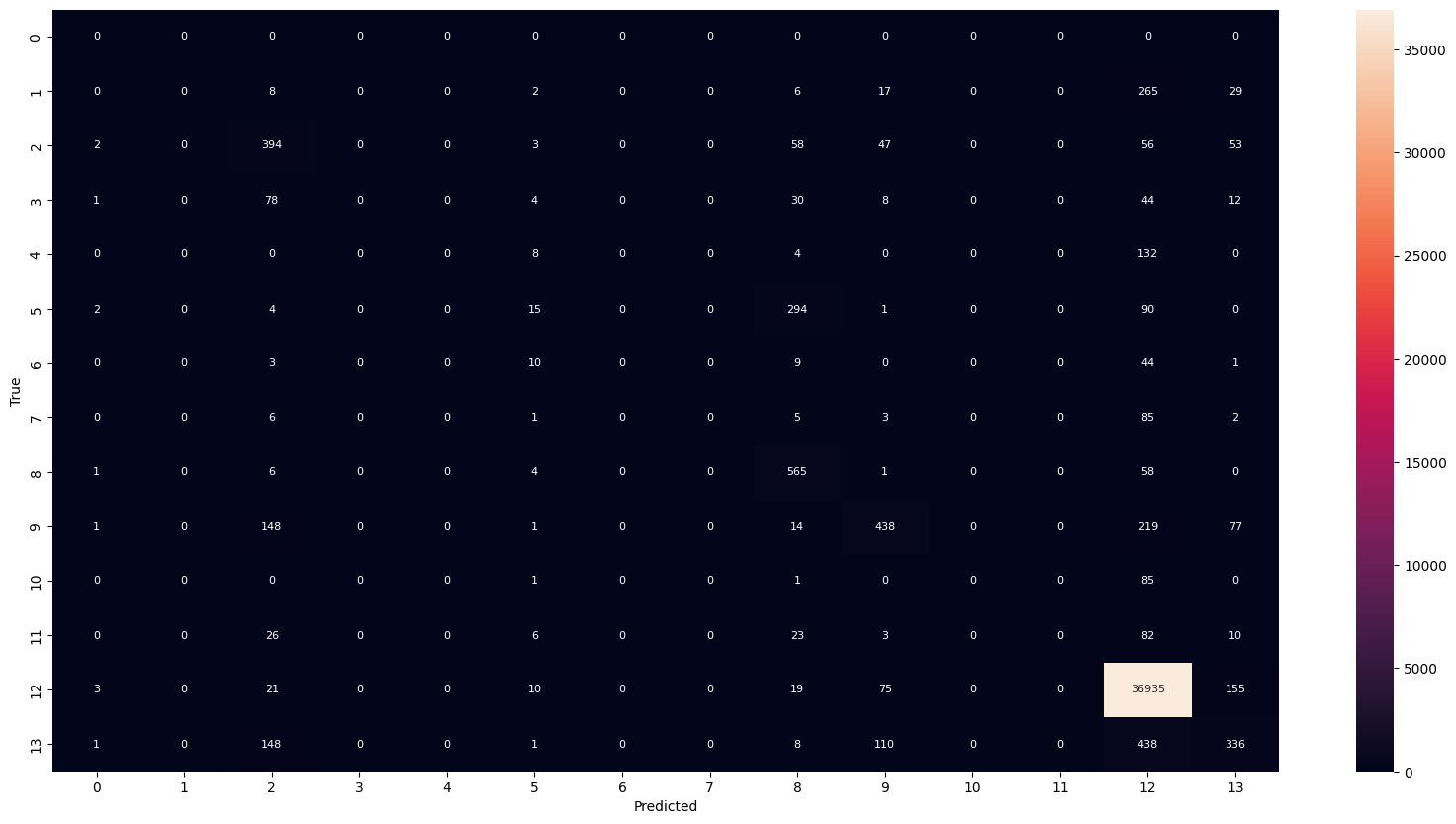
## بخش د

A screen shot of a computer code

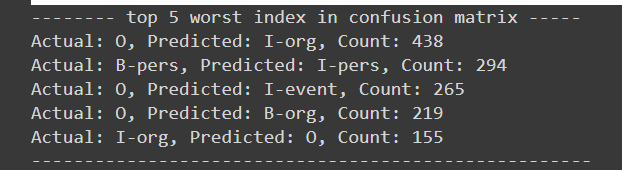
Description automatically generated

از کد بالا برای رسم بهترین ماتریس درهمریختگی بهترین مدل های بخش ب و ج رسم می­کند همچنین 5 تا از بدترین نتیجه های هر دو مدل را نمایش می­دهد.

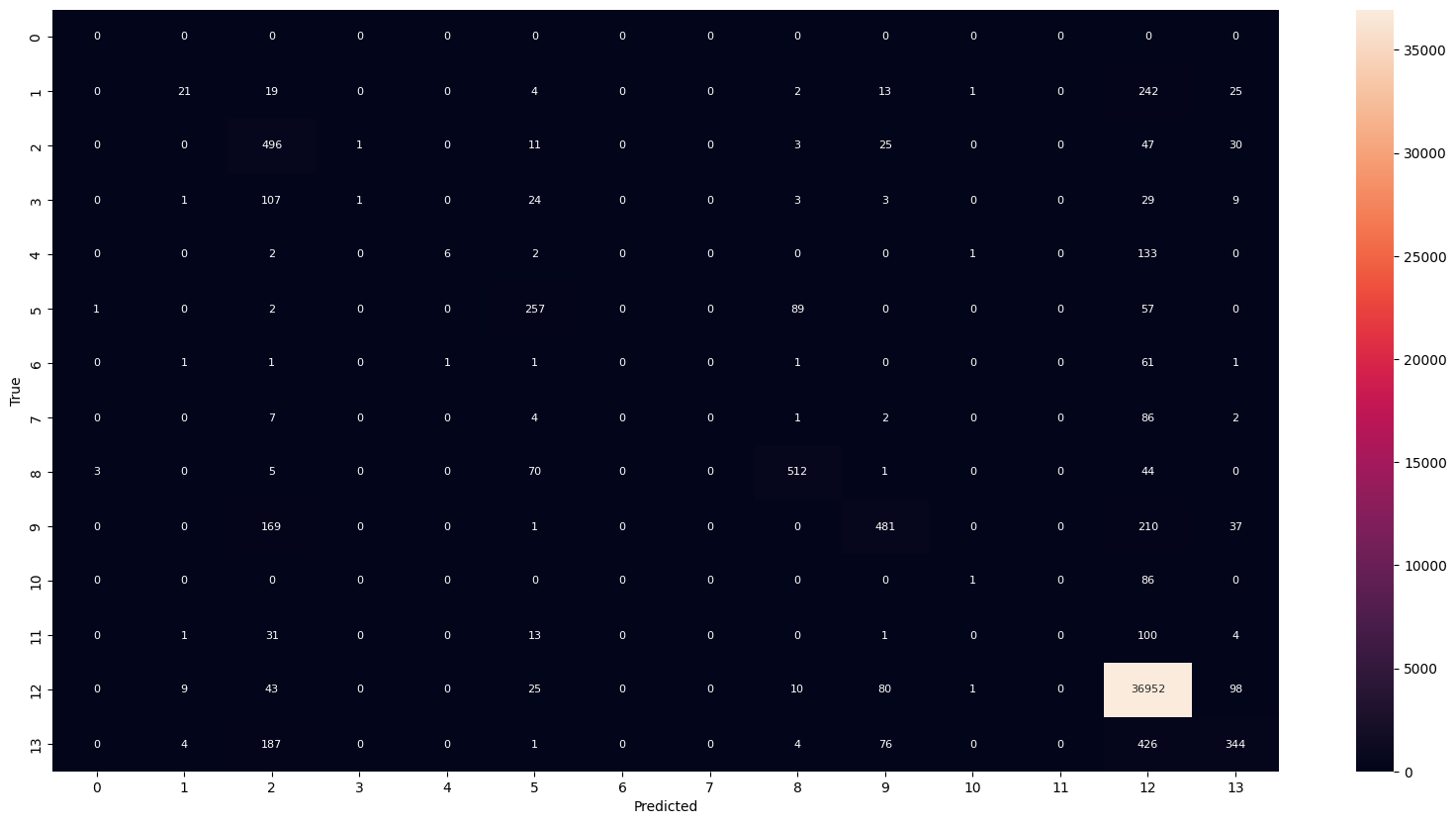
مدل بخش ب:



در لیست پراشتباه ترین موارد در ماتریس درهم­ریختگی برای بهترین مدل بخش ب:



مدل بخش ج:



در لیست پراشتباه ترین موارد در ماتریس درهم­ریختگی برای بهترین مدل بخش ج:

A screen shot of a computer

Description automatically generated

انالیز نتایج:

- مدل CRF-LSTM عملکرد بهتری نشان می‌دهد، با اشتباهات کمتر و نرخ تشخیص درست بیشتر این را با مقایسه نتیجه به دست آمده از نتایج در سطح entity متوجه شدم.

خروجی در سطح emtity برای مدل بخش ب:

A screenshot of a computer screen

Description automatically generated

خروجی در سطح emtity برای مدل بخش ج:

A screen shot of a computer

Description automatically generated

-

هر دو مدل در برچسب‌های خاصی اشتباهات رایج یکسانی دارند.

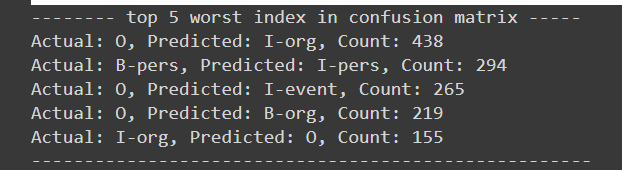
A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

- برچسب‌هایی مانند O , B-Pers , B-org به طور مداوم توسط هر دو مدل به طور متوسط به نظر بهتر تشخیص داده می‌شوند، که نشان می‌دهد این دسته‌ها برای مدل‌ها آسان‌تر هستند.

A screen shot of a computer

Description automatically generated

- برچسب‌های که تعداد بالاتری دارند و خطای آن ها در عملکرد مدل بسیار حساس است نرخ خطای بالاتری دارند. چیزی که مشخص است بیش ترین موردی که مدل با آن مشکل دارد این تگ هاست درصد اشتباه آن به نسبت تعداد آن بالاست.

- چیزی که مشخص است تگ O به علت تعداد زیاد باعث می­شود عملکردی دروغین دریافت کنیم که شاید دقت آن بالا باشد اما نباید به آن توجه کرد. شاید بهتر باشد از ارزیابی سطح entity برای خروجی اخر این سیستم استفاده کرد.

# بخش 4

در تسک NER، ارزیابی می‌تواند در دو سطح مختلف انجام شود: سطح token و سطح level. هر دو سطح دیدگاه‌های متفاوتی از عملکرد مدل NER ارائه می‌دهند.

## سطح token

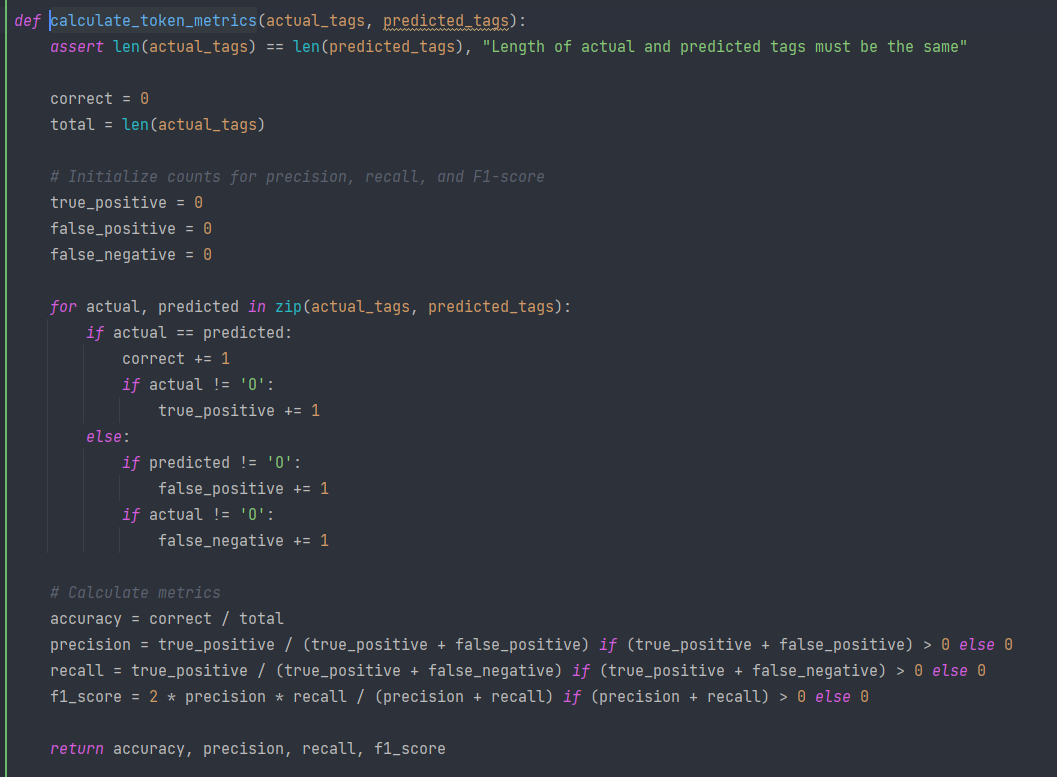
در سطح token، ارزیابی هر token کلمه را به صورت مجزا در نظر می‌گیرد. این به این معنی است که برای هر کلمه در متن، برچسب پیش‌بینی‌شده با برچسب واقعی مقایسه می‌شود.

معیارها:

- دقت Accuracy: نسبت token‌های درست پیش‌بینی‌شده به کل تعداد token‌ها.

- دقت، بازخوانی، امتیاز F1: بر اساس تعداد token‌های درست پیش‌بینی‌شده برای هر برچسب محاسبه می‌شود.

با استفاده تابع پایین در ارزیابی در سطح توکن انجام دادم.



نتایج برای مدل قسمت ب:

A screen shot of a computer

Description automatically generated

نتایج برای قسمت ج:

A screenshot of a computer screen

Description automatically generated

## سطح entity

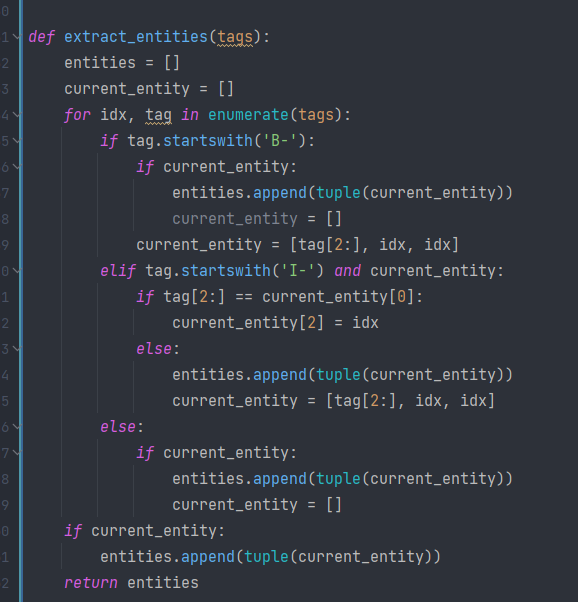
در سطح entity، ارزیابی کل entity‌ها را به صورت یک واحد در نظر می‌گیرد. یک entity به درستی تشخیص داده می‌شود فقط اگر همه token‌های تشکیل‌دهنده آن به درستی برچسب‌گذاری شده باشند. این روش سخت‌گیرانه‌تر است زیرا یک پیش‌بینی ناقص که فقط برخی token‌های entity به درستی تشخیص داده شده‌اند به عنوان موفقیت در نظر گرفته نمی‌شود.

معیارها:

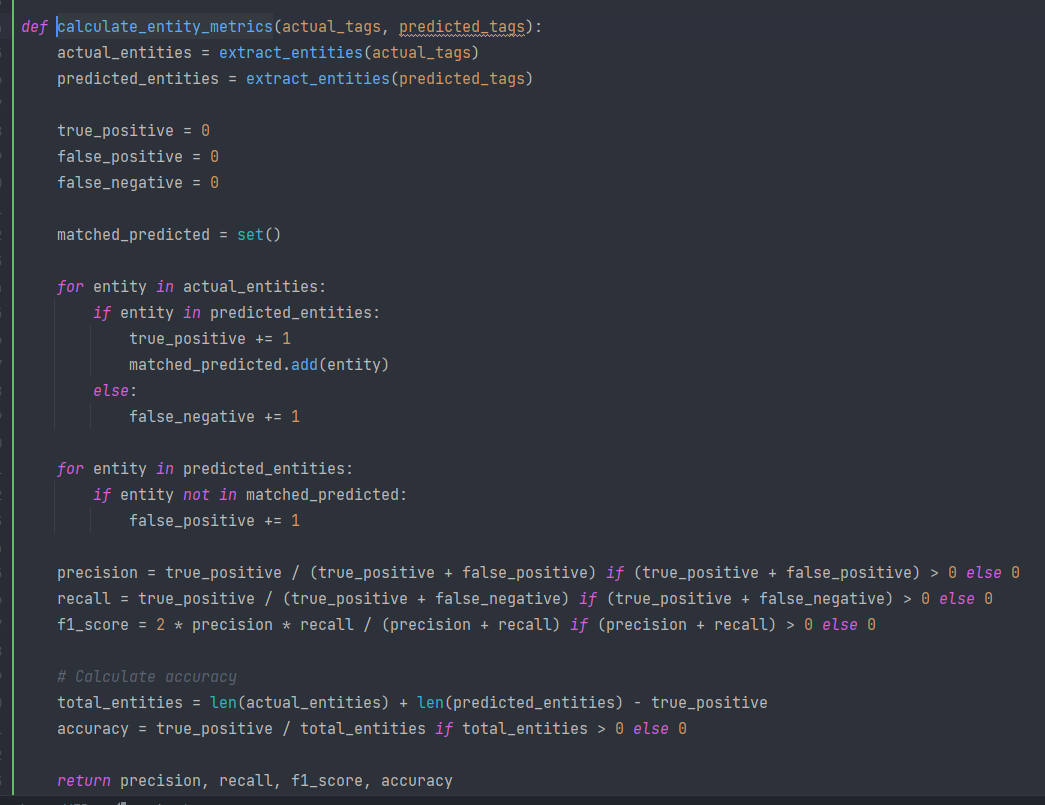
- دقت، بازخوانی، امتیاز F1: بر اساس تعداد entity‌های درست پیش‌بینی‌شده محاسبه می‌شود. یک entity به درستی پیش‌بینی‌شده در نظر گرفته می‌شود اگر و فقط اگر همه token‌های آن به درستی برچسب‌گذاری شده باشند.

با استفاده از توابع زیر این بخش پیاده­سازی کردم:

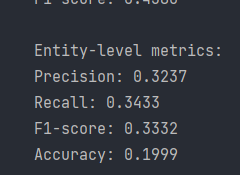
تایع زیر یک entity را به صورت کامل برای ما استخراج می­کند.



در نهایت با تابع زیر تمام entity ها را استخراج می­کنیم و عملکرد را در این سطح بیان می­کنیم.



خروجی در سطح emtity برای مدل بخش ب:



خروجی در سطح emtity برای مدل بخش ج:

A screen shot of a computer

Description automatically generated

که مشاهده می­شود با crf نتایج به طور قابل توجهی بهتر شده.

- سطح token: token‌های جداگانه را ارزیابی می‌کند، منجر به امتیازات بالاتر می‌شود اگر اکثر token‌ها به درستی برچسب‌گذاری شده باشند، حتی اگر entity‌ها به طور کامل تشخیص داده نشوند.

- سطح entity: کل entity‌ها را ارزیابی می‌کند، معیار سخت‌گیرانه‌تر و اغلب واقع‌بینانه‌تری از عملکرد مدل در کاربردهای دنیای واقعی ارائه می‌دهد.

- موارد استفاده:

- سطح token: مفید برای درک عملکرد مدل در مقیاس دقیق‌تر، به ویژه در طول فرایند آموزش.

- سطح entity: برای ارزیابی انتها به انتهای سیستم‌های NER معنی‌دارتر است، زیرا توانایی مدل در تشخیص کامل entity‌ها را منعکس می‌کند.