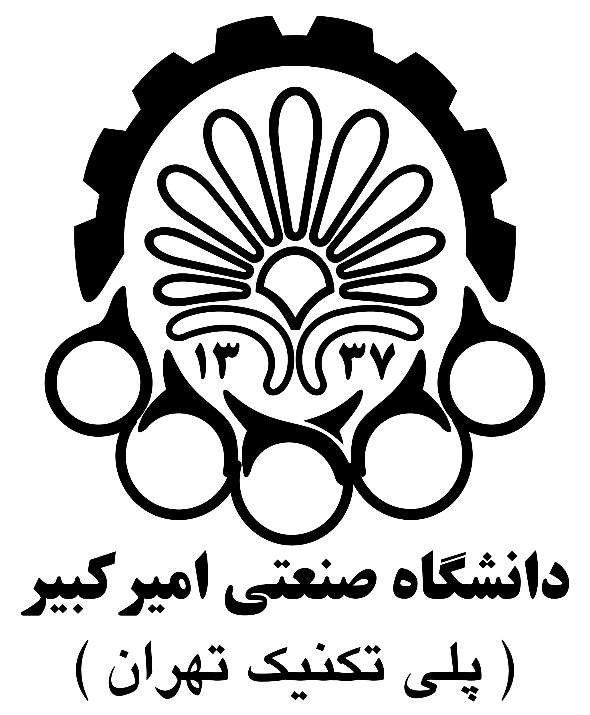
****

**دانشکده مهندسی کامپیوتر**

**تمر ین دوم درس فهم زبان**

**دکتر زینلی**

**غلامرضا دار 400131018**

**بهار 1401**

فهرست مطالب

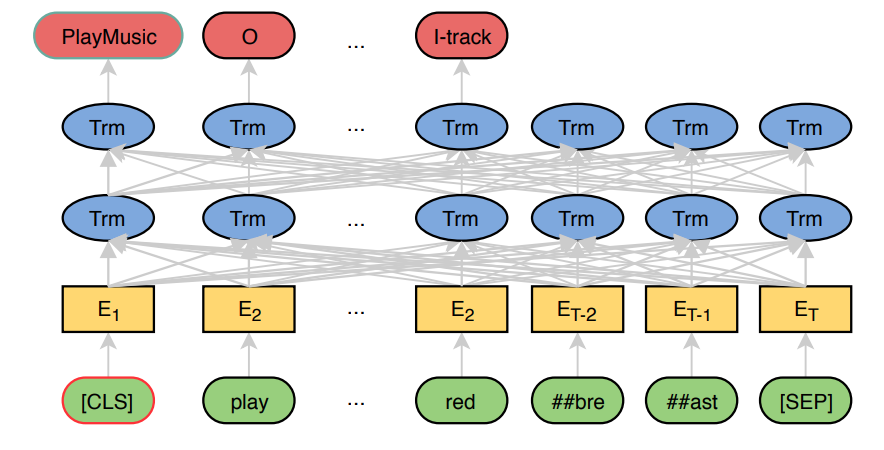
[1) جوینت برت 3](#_Toc121587447)

[2) انکدر دیکدر با اتنشن 6](#_Toc121587448)

[مراجع 9](#_Toc121587449)

# 1) جوینت برت

مدل انتخابی اول برای حل این مسئله مدل JointBert از مقاله [BERT for Joint Intent Classification and Slot Filling](https://arxiv.org/abs/1902.10909) است. معماری کلی این مدل را می­توانید در شکل زیر مشاهده کنید.



این مدل با گرفتن دنباله ورودی(توکنایزشده برای BERT)، به طور همزمان، Intent و تگ­های لازم برای هر توکن را مشخص می­کند. به ابتدای هر جمله ورودی، یک توکن [CLS] و به انتهای هر جمله یک توکن [SEP] اضافه می­کنیم. مدل با کمک این دو توکن می­تواند حدود یک جمله را مشخص کند.

خروجی مربوط به ورودی توکن [CLS]، Intent تشخیص داده شده است. همچنین، خروجی مربوط به اولین زیرتوکن هر توکنِ میانی، tag آن توکن را مشخص می­کند.

پس از انجام تغییرات لازم در این پیاده سازی و آماده سازی مجموعه داده سوال برای این مدل، شروع به آموزش مدل می­کنیم.

پیاده­سازی این مدل در این آدرس موجود است: <https://github.com/monologg/JointBERT>

**نکته در مورد مجموعه داده:** پس از بررسی مجموعه داده این سوال، متوجه شدیم که تعداد زیادی از دادگان تکراری بودند. تقریبا هر داده 5 بار کپی شده بود. این اتفاق به نظر سهوی می­آمد در نتیجه قبل از شروع به آموزش، دادگان تکراری حذف شدند.

**نصب ماژول­های مورد نیاز:**

* python>=3.7
* torch==1.5.1
* seqeval==1.2.2
* transformers==4.3.0
* pytorch-crf==0.7.2

**راهنمایی آموزش مدل:**

درون پوشه اصلی کد­ها یک فایل به نام main.py وجود دارد. پس از اطمینان از نصب بودن ماژول­های ذکرشده و دسترسی به cuda، با استفاده از دستور زیر آموزش مدل را شروع کنید.*(دقت کنید در ترمینال، در دایرکتوری ای باشید که کد main.py قرار دارد).*

$ python **main.py** --data\_dir data --model\_dir output/test\_model --task **nlu** --model\_type **joint\_bert** --do\_train

*حدود* ***40 دقیقه*** *در محیط گوگل کولب به زمان نیاز دارد*

**راهنمایی تست مدل:**

درون پوشه اصلی کد­ها یک فایل به نام main.py وجود دارد. **پس از اتمام آموزش** برای ارزیابی، دستور زیر را اجرا کنید.*(دقت کنید در ترمینال، در دایرکتوری ای باشید که کد main.py قرار دارد)*

$ python **main.py** --data\_dir data --model\_dir output/test\_model --task **nlu** --model\_type **joint\_bert** --do\_eval

*حدود* ***40 ثانیه*** *در محیط گوگل کولب به زمان نیاز دارد*

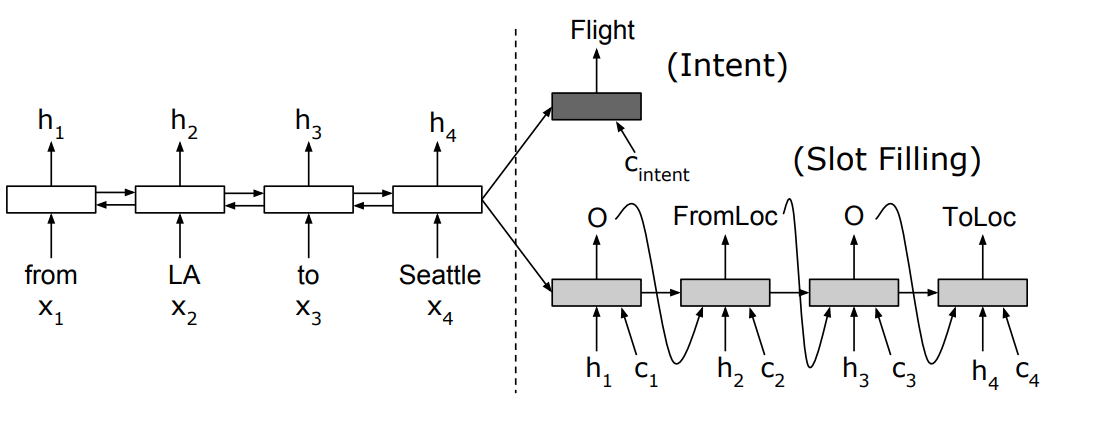
**نتایج:** همان­طور که انتظار می­رفت نتایج حاصل از استفاده از این روش بسیار مطلوب است. صحت تشخیص Intent 99 درصدی و slot\_f1 95 درصدی بر روی داده تست نتایج بسیار قابل قبولی برای این مسئله هستند. از آنجایی که جمله­های مجموعه داده، جمله­های کوتاه و ساده­ی انگلیسی بودند و BERT بر روی این نوع داده­ها به خوبی آموزش دیده است، تشخیص این­که هدف هر کدام ازاین جمله­ها چیست برای BERT کار ساده ای است. همچنین ساختار BERT اجازه می­دهد مدل در بخش Slot Filling، به خوبی بر روی جمله ورودی تسلط داشته باشد و کلمات مختلف متن را از نظر کاربرد نیز تحلیل کند.

|  |  |
| --- | --- |
| JointBERT[1] (Test) | |
| 0.1525 | eval\_loss |
| 0.9937 | intent\_acc |
| 0.9581 | Slot\_f1 |
| 0.9598 | Slot\_recall |
| 0.9565 | Slot\_precision |
| 0.9879 | Semantic\_frame\_acc |

|  |  |
| --- | --- |
| JointBERT[1] (Development) | |
| 0.1657 | eval\_loss |
| 0.9892 | intent\_acc |
| 0.9585 | Slot\_f1 |
| 0.9631 | Slot\_recall |
| 0.9539 | Slot\_precision |
| 0.9838 | Semantic\_frame\_acc |

# 2) انکدر دیکدر با اتنشن

مدل انتخابی دوم برای حل این مسئله مدل Encoder Decoder with Attention از مقاله [Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling](https://arxiv.org/abs/1609.01454) است. معماری کلی این مدل را می­توانید در شکل زیر مشاهده کنید.



بخش انکدر این مدل یک شبکه Bidirectional LSTM است. توکن­های ورودی به این شبکه داده می­شوند و خروجی سلول آخر به عنوان ورودی به Classifier داده می­شود تا Intent جمله محاسبه شود. در ادامه Hidden State آخرین سلول LSTM بازگشتی، به عنوان Initial State به اولین سلول LSTM دیکدر داده می­شود.دیکدر در این مدل یک LSTM یک­جهته است. همان­طور که در شکل مشخص است، خروجی سلول­های دیکدر در هر مرحله زمانی، tag های تشخیص داده شده برای توکن ورودی در همان مرحله زمانی را مشخص می­کنند. های موجود در بخش Decoder در واقع **مکانیزم** **اتنشن** در این مدل هستند و مدل از این طریق در هر مرحله زمانی، می­تواند با استفاده از Cها به سایر نقاط جمله توجه کند و اطلاعات لازم را کسب کند.

پس از انجام تغییرات لازم در این پیاده سازی و آماده سازی مجموعه داده سوال برای این مدل، شروع به آموزش مدل می­کنیم.

پیاده­سازی این مدل در این آدرس موجود است: <https://github.com/yinghao1019/Joint_learn>

**نکته در مورد مجموعه داده:** پس از بررسی مجموعه داده این سوال، متوجه شدیم که تعداد زیادی از دادگان تکراری بودند. تقریبا هر داده 5 بار کپی شده بود. این اتفاق به نظر سهوی می­آمد در نتیجه قبل از شروع به آموزش، دادگان تکراری حذف شدند.

**نتایج:** در مورد این مدل نیز نتایج بسیار قابل­قبول هستند. تقریبا در تمامی معیار­ها مدل بر اساس BERT عملکرد بهتری داشت اما با این وجود نتایج مدل Encoder Decoder with Attention نیز بسیار خوب هستند.

|  |  |
| --- | --- |
| EncDecAttn[2] (Test) | |
| 0.2000 | eval\_loss |
| 0.9879 | intent\_acc |
| 0.9431 | Slot\_f1 |
| 0.9381 | Slot\_recall |
| 0.9420 | Slot\_precision |
| 0.9722 | Semantic\_frame\_acc |

|  |  |
| --- | --- |
| EncDecAttn [2] (Development) | |
| 0.2143 | eval\_loss |
| 0.9804 | intent\_acc |
| 0.9401 | Slot\_f1 |
| 0.9381 | Slot\_recall |
| 0.9420 | Slot\_precision |
| 0.9722 | Semantic\_frame\_acc |

**نصب ماژول­های مورد نیاز:**

* python>=3.7
* torch==1.5.1
* seqeval==1.2.2
* transformers==4.3.0
* pytorch-crf==0.7.2

**راهنمایی آموزش مدل:**

درون پوشه اصلی کد­ها یک فایل به نام main.py وجود دارد. پس از اطمینان از نصب بودن ماژول­های ذکرشده و دسترسی به cuda، با استفاده از دستور زیر آموزش مدل را شروع کنید.*(دقت کنید در ترمینال، در دایرکتوری ای باشید که کد main.py قرار دارد)*

$ python **main.py** --data\_dir data --model\_dir output/test\_model\_s2s --task **nlu** --model\_type **joint\_AttnS2S** --do\_train

*حدود* ***8 دقیقه*** *در محیط گوگل کولب به زمان نیاز دارد*

**راهنمایی تست مدل:**

درون پوشه اصلی کد­ها یک فایل به نام main.py وجود دارد. پس از اتمام آموزش برای ارزیابی، دستور زیر را اجرا کنید.*(دقت کنید در ترمینال، در دایرکتوری ای باشید که کد main.py قرار دارد)*

$ python **main.py** --data\_dir data --model\_dir output/test\_model\_s2s --task **nlu** --model\_type **joint\_AttnS2S** --do\_eval

*حدود* ***20 ثانیه*** *در محیط گوگل کولب به زمان نیاز دارد*

* اگر در اجرای کد­ها به مشکل خوردید می­توانید نوت­بوک فراهم­شده­ای که تمام­نکات گفته شده را انجام می­دهد را در گوگل کولب آپلود کنید و در آن صورت نباید به مشکلی بر بخورید.
* اسم نوتبوک : NLU\_HW02\_colab\_nb.ipynb

# مراجع

[1] Chen, Qian, Zhu Zhuo, and Wen Wang. "**Bert for joint intent classification and slot filling.**" *arXiv preprint arXiv:1902.10909* (2019).

[2] Liu, Bing, and Ian Lane. "**Attention-based recurrent neural network models for joint intent detection and slot filling.**" *arXiv preprint arXiv:1609.01454* (2016).