به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی برق

یادگیری عمیق

پروژه نهایی

عنوان:

GAN-BERT

اعضای گروه:

آرمین قوجه‌زاده

پوریا دادخواه

عطیه میرزائی

استاد درس:

دکتر فاطمی‌زاده

بهمن ۱۴۰۲

فهرست مطالب

فصل ۱

مقدمه

امروزه روش‌های یادگیری عمیق بسیاری در پردازش زبان‌های طبیعی به کار گرفته شده‌اند. یکی از این روش‌ها استفاده از ساختارهای ترنسفورمری مانند BERT[۱] است که برای محاسبه نمایش برداری مناسب برای داده‌ها به کار می‌رود. این ساختارها اصولا بر روی مجموعه داده‌های بزرگی آموزش می‌بینند و سپس برای کاربردهای مختلف تنظیم دقیق می‌شوند تا به دقت بالاتری برسند. این دقت بالا زمانی حاصل می‌شود که این مدل‌ها بر روی هزاران داده برچسب‌دار آموزش داده شوند.

دستیابی به این حجم از داده‌های برچسب‌دار فرآیندی پیچیده و زمانبر است؛ به همین دلیل روش‌های نیمه‌نظارتی زیادی گسترش پیدا کرده‌اند. در این روش‌ها فرض می‌شود داده‌های برچسب‌دار محدودی در اختیار داریم ولی امکان دستیابی به داده‌های بدون برچسب نیز وجود دارد. یکی از این ساختارها SS-GANها هستند که شامل دو بخش شبکه مولد و شبکه تمییزدهنده می‌شوند. شبکه مولد برای تولید نمونه‌هایی مشابه داده‌های اصلی و شبکه تمییزدهنده برای تمایز دادن نمونه‌های مولد از داده‌های واقعی آموزش می‌بینند. البته تمییزدهنده علاوه بر تشخیص جعلی یا واقعی بودن نمونه‌ها، آن‌ها را در چند کلاس نیز طبقه‌بندی می‌کند.

در ادامه SS-GANها، ساختار GAN-BERT[۲]، که تعمیمی از مدل BERT در حالت مولد تخاصمی نیمه‌نظارتی است، و همچنین دیتاست مورد استفاده در این پروژه را تشریح می‌کنیم.

۱ـ۱ SS-GAN

تمییزدهنده D در SS-GANها بر روی $k+1$ کلاس آموزش می‌بینند: نمونه‌های واقعی در یکی از کلاس‌های $(1,2,...,k)$ و نمونه‌خای جعلی در $k+1$ کلاس طبقه‌بندی می‌شوند. به عبارت بهتر، فرض کنید p_d و p_G به ترتیب توزیع احتمال نمونه‌های واقعی و جعلی تولید شده باشد. همچنین فرض کنید $p_m(\hat{y}=y|x,y\in(1,...,k))$ نشانگر احتمال جعلی بودن نمونه x و $p_m(\hat{y}=y|x,y=k+1)$ نشانگر احتمال واقعی بودن نمونه (که در نتیجه متعلق به یکی از k کلاس اول باشد) برای مدل m باشد. در این صورت تابع هزینه تمییزدهنده بصورت زیر تعریف می‌شود:

$$L_D = L_{D_{sup}} + L_{D_{unsup}} \tag{۱ـ۱}$$

که در آن داریم:

$$L_{D_{sup}} = -\mathbb{E}_{x,y\sim p_d}\log[p_m(\hat{y}=y|x,y\in(1,...,k))] \tag{۱ـ۲آ}$$

$$\begin{aligned} L_{D_{unsup}} = &-\mathbb{E}_{x\sim p_d}\log[1-p_m(\hat{y}=y|x,y=k+1)] \\ &-\mathbb{E}_{x\sim G}\log[p_m(\hat{y}=y|x,y=k+1)] \end{aligned} \tag{۱ـ۲ب}$$

$L_{D_{sup}}$ خطای تخصیص نمونه واقعی به کلاس نادرست (که یکی از k کلاس اول است) را اندازه می‌گیرد و $L_{D_{unsup}}$ نیز خطای تشخیص نمونه واقی بدون برچسب به عنوان نمونه جعلی را در نظر می‌گیرد.

بطور همزمان انتظار می‌رود که شبکه مولد G بتواند نمونه‌هایی شبیه به نمونه‌های واقعی با توزیع p_d تولید کند. به عبارت دقیقتر متوسط نمونه‌های تولید شده توسط شبکه مولد باید نزدیک به مقدار متناظرش در داده‌های واقعی باشد. فرض کنید $f(x)$ نشان‌دهنده خروجی تابع فعالیت در یکی از لایه‌های میانی D باشد؛ در این حالت تابع هزینه feature matching برای G بصورت زیر تعریف می‌شود:

$$L_{G_{feature\ matching}} = ||\mathbb{E}_{x\sim p_d}f(x) - \mathbb{E}_{x\sim G}f(x)||_2^2 \tag{۱ـ۳}$$

همچنین لازم است تابع هزینه دیگری برای مولد در نظر گرفته شود تا خطای ناشی از تشخیص داده‌های جعلی توسط تمییزدهنده را محاسبه کند:

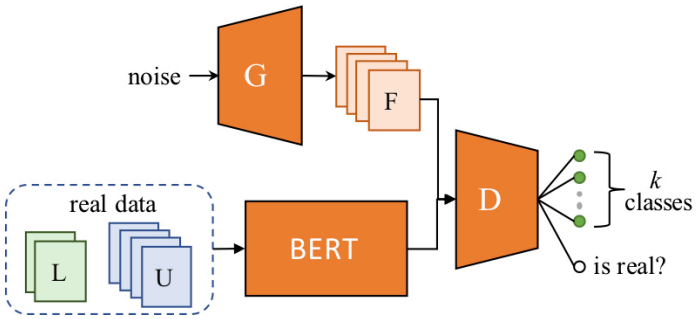
$$L_{G_{unsup}} = -\mathbb{E}_{x\sim G}\log[1-p_m(\hat{y}=y|x,y=k+1)] \tag{۱ـ۴}$$

به این ترتیب تابع هزینه شبکه مولد برابر است با

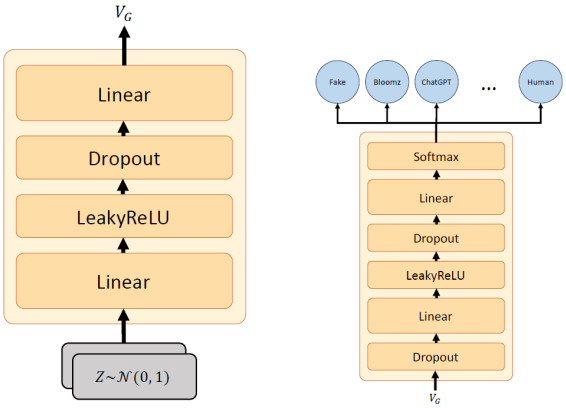
$$L_G = L_{G_{feature\ matching}} + L_{G_{unsup}}. \tag{۱ـ۵}$$

GAN-BERT ۱ـ۲

در ساختار GAN-BERT که در شکل ۱ـ۱ نمایش داده شده است، تمییزدهنده D وظیفه طبقه‌بندی هر نمونه به یکی از k کلاس را دارد. همچنین باید تشخیص دهد که داده واقعی است و یا توسط مولد تولید شده است (مجموعا $k+1$ کلاس). شبکه مولد G نیز وظیفه تولید نمایش برداری مشابه نمایش برداری داده‌های واقعی را برعهده دارد. ساختار هر یک از این دو شبکه در شکل ۱ـ۲ نمایش داده شده است.

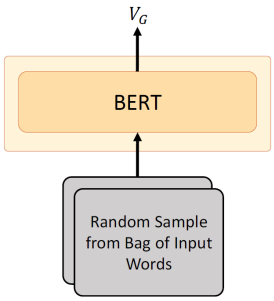


شکل ۱ـ۱: معماری GAN-BERT



شکل ۱ـ۲: معماری شبکه D (سمت راست) و G_1 (سمت چپ)

می‌توان معماری دیگری نیز برای شبکه مولد استفاده کرد که در شکل ۱ـ۳ نمایش داده شده است. در این معماری از یک شبکه از پیش آموزش‌دیده BERT استفاده می‌شود که مستقل از BERT استفاده شده در شکل ۱ـ۱ است. در این روش ابتدا یک bag of words از کلمات حاضر در نمونه‌های آموزشی ساخته می‌شود . سپس بر اساس توزیع این کلمات، نمونه‌های تصادفی $\tilde{X}$ بصورت i.i.d. انتخاب و به BERT داده می‌شوند.



شکل ۱ـ۳: معماری شبکه G_2

۱ـ۳ دیتاست

دیتاست این پروژه تعمیمی از دیتاست M4 است که مشخصات آن در جدول ۱ـ۱ نمایش داده شده است. این دیتاست به فرمت json است و هر نمونه از موارد زیر تشکیل شده است:

- id: شناسه نمونه
- label: کلاس نمونه (کلاس‌ها: human, ChatGPT, Cohere, Davinci, Bloomz, Dolly)
- text: متنی که قرار است طبقه‌بندی شود
- model: مدلی که متن توسط آن تولید شده
- source: منبع نمونه به دست آمده شامل Wikipedia, Wikihow, Peerread, Reddit, Arxiv

جدول ۱ـ۱: مشخصات دیتاست

Source/ Domain	Language	Total Human	Parallel Data						
			Human	Davinci003	ChatGPT	Cohere	Dolly-v2	BLOOMz	Total
Wikipedia	English	6,458,670	3,000	3,000	2,995	2,336	2,702	3,000	17,033
Reddit ELI5	English	558,669	3,000	3,000	3,000	3,000	3,000	3,000	18,000
WikiHow	English	31,102	3,000	3,000	3,000	3,000	3,000	3,000	18,000
PeerRead	English	5,798	5,798	2,344	2,344	2,344	2,344	2,344	17,518
arXiv abstract	English	2,219,423	3,000	3,000	3,000	3,000	3,000	3,000	18,000
Baike/Web QA	Chinese	113,313	3,000	3,000	3,000	–	–	–	9,000
RuATD	Russian	75,291	3,000	3,000	3,000	–	–	–	9,000
Urdu-news	Urdu	107,881	3,000	–	3,000	–	–	–	9,000
id_newspapers_2018	Indonesian	499,164	3,000	–	3,000	–	–	–	6,000
Arabic-Wikipedia	Arabic	1,209,042	3,000	–	3,000	–	–	–	6,000
True & Fake News	Bulgarian	94,000	3,000	3,000	3,000	–	–	–	9,000
Total			35,798	23,344	32,339	13,680	14,046	14,344	133,551

در این پروژه $k = 6$ است و هدف طبقه‌بندی داده‌ها بر اساس متن ورودی بر هر یک از کلاس‌ها است. برای این کار هر بار درصدهای مختلفی از دیتاست را به عنوان داده‌های برچسب‌دار در نظر میگیریم و به تدریج تعداد این نمونه‌ها را زیاد و نتایج را مقایسه و گزارش می‌کنیم.

فصل ۲

خواسته اول

در این قسمت مدل BERT را با هدف طبقه‌بندی متون به کلاس‌های مختلف fine-tune می‌کنیم. در ادامه توضیح مفصلی راجع به نحوه پیاده‌سازی این بخش و نتایج آن خواهیم داشت.

۲ـ۱ تشریح روش

گام‌های پیاده‌سازی به این شرح هستند:

۱. فراخوانی کتابخانه‌ها: در این بخش کتابخانه‌های مختلفی از جمله موارد مربوط به دیتاست (با استفاده از کتابخانه‌های Hugging Face)، تغییرات داده‌ها (Pandas و NumPy) و سایر کتابخانه‌ها (مانند scikit-learn و Matplotlib) را فراخوانی می‌کنیم.

۲. تنظیم seed و دستگاه

۳. لود کردن دیتاست و تقسیم آن به بخش‌های train، test و validation

۴. پیش‌پردازش: از BERT tokenizer برای توکن‌سازی داده‌های آموزش و اعتبارسنجی استفاده می‌شود.

۵. مقداردهی اولیه مدل: مدل طبقه‌بند مبتنی بر BERT (`BertForSequenceClassification`) مقداردهی اولیه می‌شود و تعداد لیبل‌های آن معین می‌شود.

۶. تعیین هایپرپارامترهای آموزش: مواردی از جمله نرخ آموزش، سایز بچ، استراتژی ارزیابی و حداکثر تعداد گام‌های آموزش (max_step) مقداردهی می‌شوند. Trainer با دیتاست‌های آموزش و اعتبارسنجی کانفیگور می‌شود. متغیر max_step همان حداکثر گام‌های (بچ‌های) آموزش است که در fine-tune اجرا می‌شود. این متغیر برای اعمال محدودیت بر فرآیند آموزش استفاده می‌شود، علی‌الخصوص زمانی که دیتاست بسیار بزرگ باشد. در واقع این متغیر علاوه بر تنظیم زمان آموزش، مانع از overfitting نیز می‌شود.

۷. آموزش مدل با تعداد مشخص ایپوک و ذخیره بهترین مدل بر اساس نتایج اعتبارسنجی

۸. نمایش نتایج: معیارهایی همچون accuracy، precision، recall و f1-score برای هر کلاس محاسبه و بر روی نمودار نمایش داده می‌شود. معنی هر یک از این معیارها به این شرح هستند:

- accuracy: نسبت تعداد نمونه‌هایی که به درستی طبقه‌بندی شده‌اند به کل نمونه‌ها
- precision: نسبت تعداد نمونه‌هایی که در هر کلاس به درستی پیش‌بینی شده‌اند به تعداد کل نمونه‌هایی که به عنوان اعضای آن کلاس پیش‌بینی شده‌اند
- recall: نسبت تعداد نمونه‌هایی که در هر کلاس به درستی پیش‌بینی شده‌اند به تعداد کل نمونه‌هایی که واقعا در آن کلاس قرار دارند
- f1-score: متوسط precision و recall

۲ـ۲ نتایج

نتایج را با در نظر گرفتن ۸۰ درصد نمونه‌ها به عنوان مجموعه آموزش و ۲۰ درصد آن‌ها برای تست بررسی می‌کنیم (با افزایش درصد آموزشی، مشخصا نتیجه بهتری خواهیم داشت!). همچنین قرار می‌دهیم max_step=50. نتایج به این ترتیب هستند:

```
There are 1 GPU(s) available.
We will use the GPU: NVIDIA GeForce RTX 3070 Laptop GPU
```

شکل ۲ـ۱: gpu

```
[11]: train_df
```

[11]:

	text	model	source	label	id
63795	We report on observations made with the Spitze...	bloomz	arxiv	4	63795
52761	The concept of dynamical 3-space is introduced...	bloomz	arxiv	4	52761
17690	The European Association for Digital Humaniti...	davinci	wikipedia	3	17690
38189	We study theoretically the emission and absorp...	bloomz	arxiv	4	38189
18742	Stranger in the City, also known as S.I.T.C, ...	davinci	wikipedia	3	18742
...	...	...	...	...	...
31671	The chiral condensate is an important order pa...	dolly	arxiv	5	31671
29500	Yes, mercenaries in the Middle Ages carried ba...	davinci	reddit	3	29500
25898	Well, that's an interesting question. Prester ...	chatGPT	reddit	1	25898
66736	Spend your time wisely. Stop comparing yoursel...	dolly	wikihow	5	66736
13891	Laboa (Basque pronunciation:[laβoβa]) is the n...	bloomz	wikipedia	4	13891

56821 rows × 5 columns

شکل ۲ـ۲: دیتاست آموزش

```
[12]: test_df
```

[12]:

	text	model	source	label	id
0	Overall, I found the paper "Machine Comprehens...	chatGPT	peerread	1	1844
1	This paper "Machine Comprehension Using Match-...	chatGPT	peerread	1	1845
2	The paper presents an end-to-end neural archit...	chatGPT	peerread	1	1846
3	This paper proposes an end-to-end neural archi...	chatGPT	peerread	1	1847
4	Title: Incorporating long-range consistency in...	chatGPT	peerread	1	1848
...	...	...	...	...	...
2995	The paper Energy-Based Spherical Sparse Coding...	dolly	peerread	5	14560
2996	Dear Author, I have reviewed your submitted pa...	dolly	peerread	5	14561
2997	Denoising Auto-Encoders (DAE) have been used i...	dolly	peerread	5	14562
2998	The paper Revisiting Denoising Auto-Encoders, ...	dolly	peerread	5	14563
2999	This paper Revisiting Denoising Auto-Encoders ...	dolly	peerread	5	14564

3000 rows × 5 columns

شکل ۲ـ۳: دیتاست تست

```
[13]: val_df
```

[13]:

	text	model	source	label	id
30773	It is really cool that there is no sound in a ...	davinci	reddit	3	30773
55574	\n\nChanging the water pump on a 2.0L 4 Cylin...	davinci	wikihow	3	55574
39797	In our work titled "Random Access Broadcast: S...	chatGPT	arxiv	1	39797
40784	In this work, we propose a scheme for continuo...	chatGPT	arxiv	1	40784
58487	Eeva-Kaarina Aronen (born July 6, 1961) is a F...	chatGPT	wikipedia	1	58487
...	...	...	...	...	...
14116	The X.28 was the first British jet aircraft to...	bloomz	wikipedia	4	14116
20596	The Crips is an alliance of street gangs which...	human	wikipedia	0	20596
54832	As with any illness, the more time you can gi...	human	wikihow	0	54832
13223	Nimbarkas are followers of the Nimbārka school...	bloomz	wikipedia	4	13223
67579	How to Tip Tips are passed along to service pr...	dolly	wikihow	5	67579

14206 rows × 5 columns

شکل ۲ـ۴: دیتاست validation

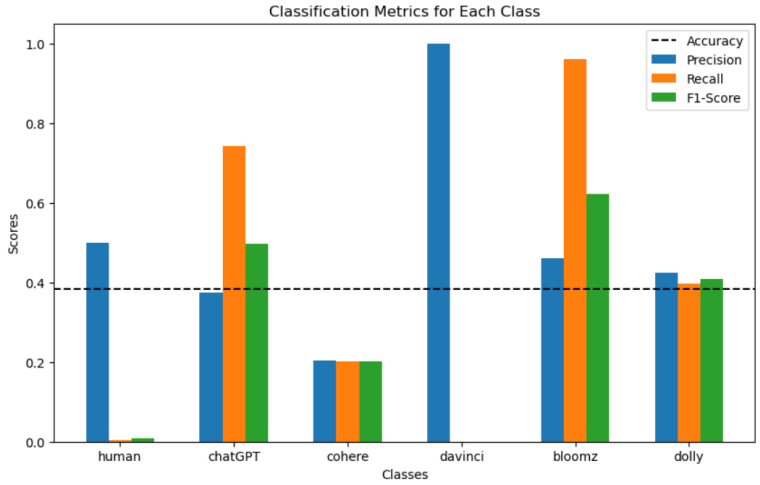
[50/50 17:00, Epoch 0/1]

Epoch	Training Loss	Validation Loss	F1 Micro
0	No log	1.658242	0.357103

شکل ۲ـ۵: نتایج دیتاست ارزیابی در روند آموزش (۱ ایپوک و max_step = 50 و ۱۷ دقیقه جهت مقایسه با سوال دو که adapter دارد)

```
human: {'precision': 0.5, 'recall': 0.004, 'f1-score': 0.007936507936507936, 'support': 500}
chatGPT: {'precision': 0.3747474747474748, 'recall': 0.742, 'f1-score': 0.4979865771812081, 'support': 500}
cohere: {'precision': 0.20321931589537223, 'recall': 0.202, 'f1-score': 0.20260782347041123, 'support': 500}
davinci: {'precision': 1.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 500}
bloomz: {'precision': 0.46065259117082535, 'recall': 0.96, 'f1-score': 0.6225680933852141, 'support': 500}
dolly: {'precision': 0.42398286937901497, 'recall': 0.396, 'f1-score': 0.4095139607032058, 'support': 500}
accuracy: 0.384
macro avg: {'precision': 0.4937670418654479, 'recall': 0.38399999999999995, 'f1-score': 0.2901021604460912, 'support': 3000}
weighted avg: {'precision': 0.49376704186544795, 'recall': 0.384, 'f1-score': 0.29010216044609116, 'support': 3000}
```

شکل ۲ـ۶: نتایج دیتاست تست



شکل ۲ـ۷: نتایج دیتاست تست بر روی نمودار

به دقت ۳۸ درصد برای یک ایپوک رسیدیم.

برای max_step=100 نیز داریم:

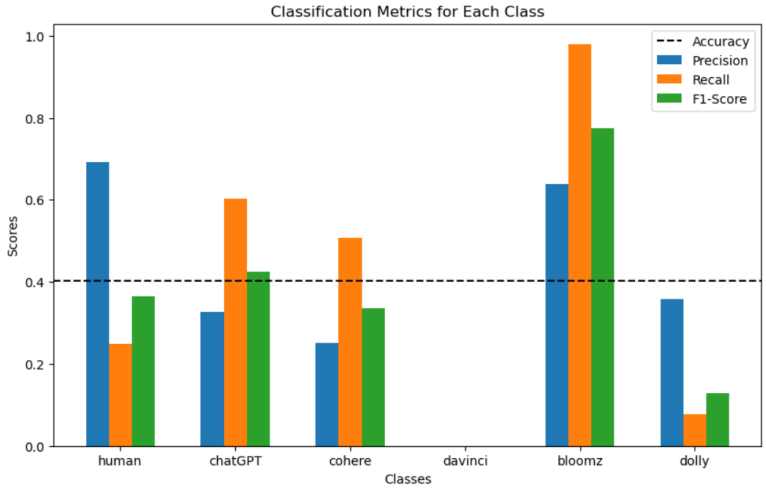
[100/100 24:10, Epoch 0/1]

Epoch	Training Loss	Validation Loss	F1 Micro
0	No log	1.480283	0.437632

شکل ۲ـ۸: نتایج دیتاست ارزیابی در روند آموزش (۱ ایپوک و max_step = 100 و ۲۴:۱۰ دقیقه)

```
human: {'precision': 0.6927374301675978, 'recall': 0.248, 'f1-score': 0.3652430044182622, 'support': 500}
chatGPT: {'precision': 0.3268398268398268, 'recall': 0.604, 'f1-score': 0.4241573033707865, 'support': 500}
cohere: {'precision': 0.2512363996043521, 'recall': 0.508, 'f1-score': 0.33620119126406356, 'support': 500}
davinci: {'precision': 0.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 500}
bloomz: {'precision': 0.639686684073107, 'recall': 0.98, 'f1-score': 0.7740916271721958, 'support': 500}
dolly: {'precision': 0.3577981651376147, 'recall': 0.078, 'f1-score': 0.12807881773399016, 'support': 500}
accuracy: 0.403
macro avg: {'precision': 0.3780497509704164, 'recall': 0.40299999999999997, 'f1-score': 0.337961990659883, 'support': 3000}
weighted avg: {'precision': 0.37804975097041643, 'recall': 0.403, 'f1-score': 0.33796199065988297, 'support': 3000}
```

شکل ۲ـ۹: نتایج دیتاست تست



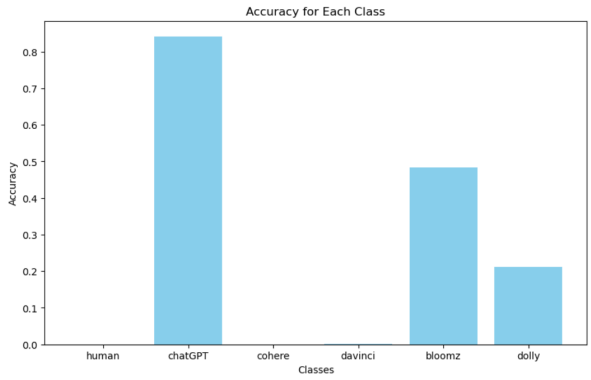
شکل ۲ـ۱۰: نتایج دیتاست تست بر روی نمودار

برای یک ایپوک از دقت ۳۸ به ۴۰ رسیدیم. همان طور که مشاهده می شود می توان مقدار `max_step` را برای رسیدن به دقت های بالاتر افزایش داد.

لازم به ذکر است ما این سوال را در حالت نیمه‌نظارتی نیز پیاده‌سازی کرده‌ایم:

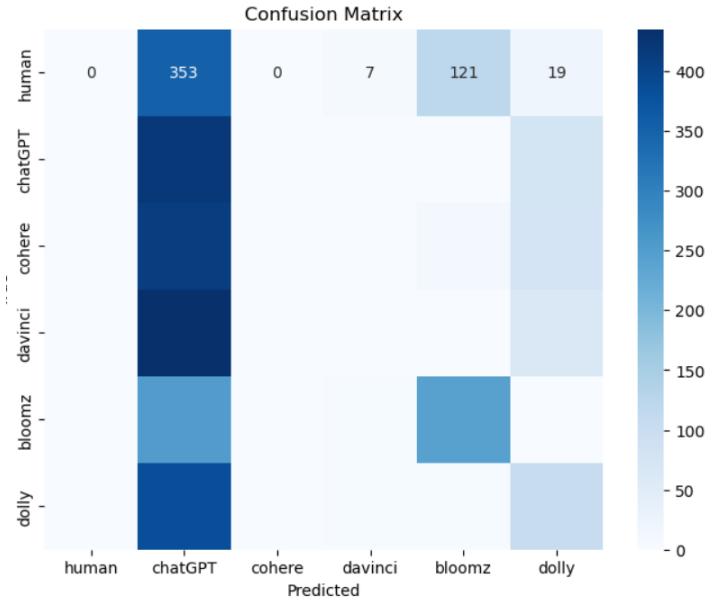
ابتدا دیتاست برچسب دار را آموزش می دهیم سپس از آن برای آموزش داده های بدون برچسب استفاده می کنیم. طبیعتا زمان به شدت بالایی برای آموزش می خواهد در کدنویسی از تکنیک self-training استفاده شده است. در این روش مدل ابتدا بر روی داده‌های برچسب‌دار آموزش می‌بیند. سپس از پیش‌بینی‌های خودش برای داده‌های بدون برچسب استفاده می‌کند تا داده‌های شبه‌برچسب‌دار تولید کند. در نهایت مدل مجددا بر روی مخلوطی از داده‌های برچسب‌دار و شبه‌برچسب‌دار آموزش می‌بیند. این کار می‌تواند تا برآورده شدن هر دقت خاصی تکرار شود. نتایج برای این روش به این شرح هستند:

```
Accuracy: 0.2567
Accuracy for class human: 0.0000
Accuracy for class chatGPT: 0.8420
Accuracy for class cohere: 0.0000
Accuracy for class davinci: 0.0020
Accuracy for class bloomz: 0.4840
Accuracy for class dolly: 0.2120
```



Classification Report:

	precision	recall	f1-score	support
human	1.00	0.00	0.00	500
chatGPT	0.19	0.84	0.31	500
cohere	1.00	0.00	0.00	500
davinci	0.06	0.00	0.00	500
bloomz	0.64	0.48	0.55	500
dolly	0.31	0.21	0.25	500
accuracy			0.26	3000
macro avg	0.53	0.26	0.19	3000
weighted avg	0.53	0.26	0.19	3000



همان گونه که مشاهده می شود دقت در حدود ۲۵ درصد برای ۲۰ درصد برچسب دار و ۸۰ درصد بدون برچسب است که زمان طولانی هم برای آموزش طی شد (۵ ساعت).

فصل ۳

خواسته دوم (امتیازی)

در این قسمت از adaptorها برای سرعت بخشیدن به فرآیند آموزش بهره می‌گیریم.

۳ـ۱ تشریح روش

کلاس Adaptor به عنوان یک لایه به مدل از پیش آموزش‌دیده BERT اضافه می‌شود. لایه adaptor در واقع یک ماژول شبکه عصبی سبک و مبتنی بر تسکی مشخص است که با هدف fine-tune یک مدل از پیش آموزش‌دیده، بدون اعمال تغییرات زیاد بر روی آن استفاده می‌شود. featureهای لایه adaptor به این شرح هستند:

- مقداردهی اولیه (`__init__`):
  - `input_size`: ابعاد لایه ورودی (در این جا ابعاد لایه مخفی BERT)
  - `output_size`: ابعاد لایه خروجی
  - `reduction_factor`: فاکتور کاهش ابعاد برای لایه‌های adaptor میانی
- لایه‌های adaptor: هر لایه adaptor از موارد زیر تشکیل شده است:
  - لایه خطی که ابعاد ورودی را متناسب با `reduction_factor` کاهش می‌دهد
  - ReLU

- لایه خطی برای مپینگ خروجی

- Forward Pass:

- متد forward تعین‌کننده forward pass برای adaptor است که بر روی هر لایه adaptor تکرار می‌شود و به تنسور ورودی اعمال می‌گردد.
- یک skip connection با اضافه کردن خروجی adaptor به ورودی اعمال می‌کند.
- skip connection موجب ذخیره اطلاعات ورودی اصلی خواهد شد

نحوه استفاده از adaptor:
کافیست یک نمونه از این کلاس را به مدل BERT اضافه کنیم. تعداد لایه‌های adaptor باید برابر با تعداد لایه‌های انکودر BERT باشد.

لایه‌های adaptor با هدف کاهش پارامترهای قابل آموزش مدل در شبکه، سریع‌تر و کاراتر ساختن فرآیند آموزش و بهبود تعمیم‌پذیری استفاده می‌شوند. تعدادی از مزایای adaptorها به این ترتیب هستند:

- کاراتر شدن پارامترهای قابل آموزش
- کارایی محاسباتی
- افزایش سرعت آموزش
- بهبود تعمیم‌پذیری
- Transfer learning مقیاس‌پذیر
- حفظ دانش مدل از پیش آموزش‌دیده

۳ـ۲ نتایج

در این قسمت از ساختار سوال اول به همراه adaptor بهره می‌گیریم. نتایج زیر به دست می‌آید:

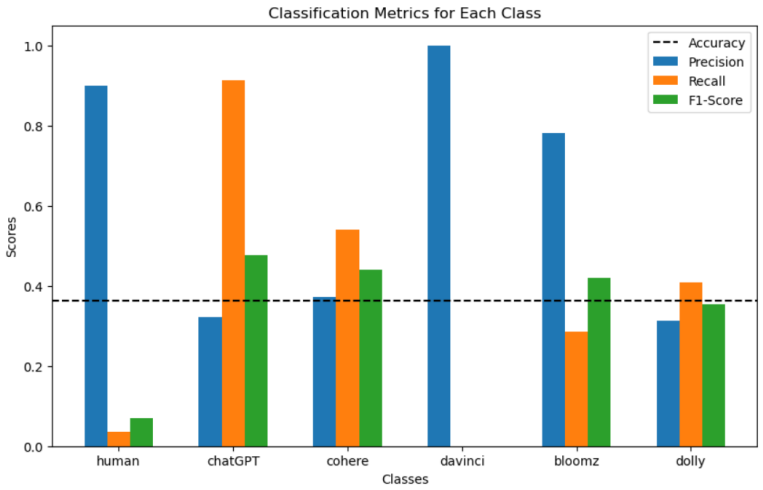
[50/50 15:47, Epoch 0/1]

Epoch	Training Loss	Validation Loss	F1 Micro
0	No log	1.679554	0.305997

شکل ۳ـ۱: نتایج دیتاست ارزیابی در روند آموزش (۱ ایپوک و $\texttt{max\_step} = 50$ و ۱۵:۴۷ دقیقه جهت مقایسه با سوال یک)

```
human: {'precision': 0.9, 'recall': 0.036, 'f1-score': 0.06923076923076922, 'support': 500}
chatGPT: {'precision': 0.32251235003528583, 'recall': 0.914, 'f1-score': 0.47678664580073027, 'support': 500}
cohere: {'precision': 0.371900826446281, 'recall': 0.54, 'f1-score': 0.4404567699836868, 'support': 500}
davinci: {'precision': 1.0, 'recall': 0.0, 'f1-score': 0.0, 'support': 500}
bloomz: {'precision': 0.7814207650273224, 'recall': 0.286, 'f1-score': 0.4187408491947291, 'support': 500}
dolly: {'precision': 0.3119266055045872, 'recall': 0.408, 'f1-score': 0.35355285961871746, 'support': 500}
accuracy: 0.364
macro avg: {'precision': 0.6146267578355794, 'recall': 0.36400000000000005, 'f1-score': 0.29312798230477216, 'support': 3000}
weighted avg: {'precision': 0.6146267578355794, 'recall': 0.364, 'f1-score': 0.29312798230477216, 'support': 3000}
```

شکل ۳ـ۲: نتایج دیتاست تست



شکل ۳ـ۳: نتایج دیتاست تست بر روی نمودار

همان طور که مشاهده می شود از ۱۷ دقیقه به ۱۵:۴۷ دقیقه رسیدیم که می‌توان با تنظیم فاکتور `reduction_factor` در adapter، زمان را بیشتر کاهش داد. همچنین به دقت ۳۶ درصد رسیدیم که در مقایسه با سوال اول، ۲ درصد کاهش داشتیم که با ساختار adapter که پارامتر های مدل را کاهش می دهد تطابق دارد.

فصل ۴

خواسته سوم

در این بخش به پیاده‌سازی GAN-BERT می‌پردازیم.

۴ـ۱ تشریح روش

پیاده‌سازی شبکه مولد در GAN-BERT با دو روش قابل انجام است که به ترتیب در شکل ۱ـ۲ (G_1) و ۱ـ۳ (G_2) نمایش داده شده است. جزئیات هر دو روش در فصل مقدمه تشریح شده است. با این حال در ادامه جزئیات پیاده‌سازی کد G_2 در مقایسه با G_1 تشریح می‌شود:
تنها کافیست موارد زیر را در کد تغییر دهیم:

`Class Generator2()`

`bag_of_words`

Instance `g2` of `Generator2` class

- در کلاس G_2، از یک مدل پیاده‌سازی BERT به نام 'bert-base-uncased' استفاده می‌کنیم و در لایه خروجی از یک لایه خطی برای تولید لیبل تولید شده توسط مولد استفاده می‌کنیم. در مرحله forward تابع، که Bag of words را در ورودی خود دریافت می‌کند، یک نمونه تصادفی از آن را گرفته و پس از Tokenize کردن توسط Tokenizer خود BertTokenizer، آن را به مدل Bert داده و از لایه خطی می‌گذرانیم.

- برای کیسه کلمات، از فایل آموزش بدون برچسب که بخش جامعی از کلمات را دارد، استفاده می‌کنیم و هر کلمه از آن را با Tokenizer معرفی شده در بخش قبلی، Tokenize می‌کنیم و به لیست `bag_of_words` اضافه می‌کنیم و لیست نهایی را به نمونه g2 از کلاس `Generator2` می‌دهیم.
- در آخر یک Instantiate از کلاس خود ساخته و در حلقه آموزش از آن به جای g1 استفاده می‌کنیم.

تحلیل دو مولد معرفی شده:

مدل مولد G_2 در مقایسه با مدل اصلی G_1 دارای چندین مزایا و معایب کلی است:

مزایا:

۱. استفاده از مدل BERT از پیش آموزش دیده: G_2 از یک مدل BERT از پیش آموزش دیده برای شبکه مولد خود استفاده می کند و به آن امکان می دهد از درک متنی و قابلیت های بازنمایی زبان BERT بهره مند شود. این می‌تواند به نمونه‌های جعلی از لحاظ معنایی معنادارتر و مرتبط‌تر منجر شود.

۲. خروجی واقعی: با استفاده از مجموعه ای از کلمات از نمونه های آموزشی برای تولید یک نمونه تصادفی، هدف G_2 تولید نمونه های جعلی واقعی تر و منسجم تر است که نماینده داده های آموزشی هستند. این به طور بالقوه می تواند کیفیت نمونه های تولید شده را بهبود بخشد.

۳. ورودی نویز ساده شده: G_2 مانند G_1 به بردار نویز سنتی متکی نیست. در عوض، از نمونه تصادفی مبتنی بر فرکانس از کیسه کلمات استفاده می کند، که ممکن است به ورودی ساختارمندتر و معنادارتری برای مولد منجر شود.

معایب:

۱. پیچیدگی: مدل G_2 پیچیدگی بیشتری را با ترکیب یک مدل BERT از پیش آموزش دیده معرفی می کند. این ممکن است منجر به افزایش نیازهای محاسباتی و زمان آموزش در مقایسه با معماری ساده‌تر G_1 شود.

۲. وابستگی به مدل از پیش آموزش دیده: اتکای G_2 به یک مدل BERT از پیش آموزش دیده آن را به کیفیت و ارتباط داده های قبل از آموزش وابسته می کند. اگر مدل BERT از پیش آموزش دیده برای کار یا دامنه خاص مناسب نباشد، ممکن است اثربخشی G_2 را محدود کند.

توضیح بیشتر سادگی و دقت:

ـ مدل G_2 لزوماً مدل کلی GAN-BERT را به دلیل پیچیدگی اضافه ادغام یک مدل BERT از پیش آموزش دیده ساده تر نمی کند. با این حال، این پتانسیل را دارد که دقت نمونه های تولید شده را با استفاده از درک متنی BERT و تولید خروجی های واقعی تر بهبود بخشد.

ـ دقت مدل G_2 در سناریوهایی که این کار مستلزم درک عمیق زبان و زمینه است، بهتر است، زیرا مدل BERT از پیش آموزش دیده می تواند الگوها و روابط پیچیده زبانی را به تصویر بکشد. با این حال، بهبود واقعی در دقت به وظیفه خاص، مجموعه داده، و کیفیت مدل BERT از پیش آموزش دیده بستگی دارد.

به طور خلاصه، مدل G_2 با ادغام یک مدل BERT از پیش آموزش‌دیده، رویکرد پیچیده‌تری را معرفی می‌کند، که به طور بالقوه می‌تواند به نمونه‌های جعلی مرتبط‌تر و واقعی‌تر منجر شود. با این حال، این با پیچیدگی بیشتر و وابستگی به کیفیت مدل از پیش آموزش دیده همراه است. تأثیر بر دقت و سادگی مدل باید به صورت تجربی در زمینه وظایف و مجموعه داده‌های خاص ارزیابی شود.

۴ـ۲ نتایج

- ابتدا نتایج را برای مدل G_1 و با در نظر گرفتن ۲۰ درصد از داده‌ها به عنوان داده برچسب‌دار و ۸۰ درصد بدون برچسب بررسی می‌کنیم:

```
  Average training loss generetor: 0.676
  Average training loss discriminator: 1.942
  Training epcoh took: 0:18:25

Running Test...
  Accuracy: 0.437
  F1 Score: 0.393
  Test Loss: 1.857
  Test took: 0:00:08
```

شکل ۴ـ۱: ایپوک ۱

```
  Average training loss generetor: 0.711
  Average training loss discriminator: 1.131
  Training epcoh took: 0:23:00

Running Test...
  Accuracy: 0.405
  F1 Score: 0.370
  Test Loss: 2.203
  Test took: 0:00:13
```

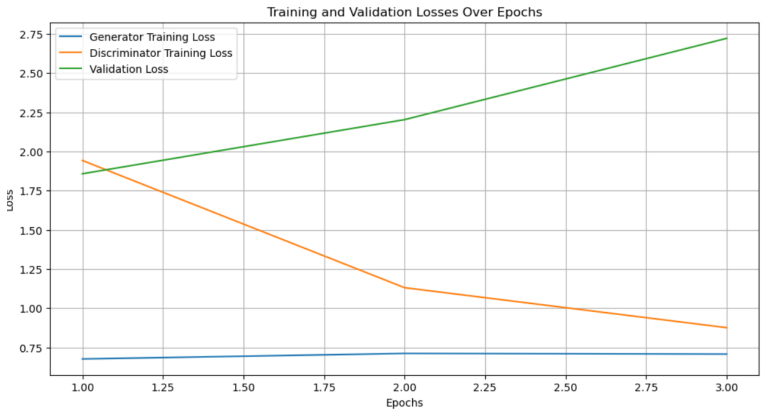
شکل ۴ـ۲: ایپوک ۲

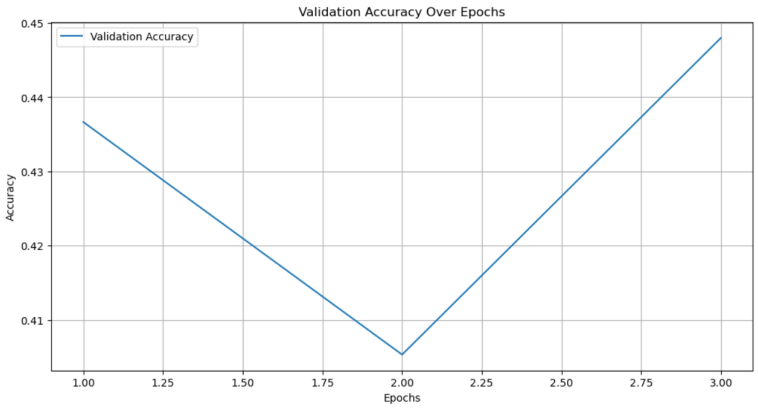
```
  Average training loss generetor: 0.707
  Average training loss discriminator: 0.876
  Training epcoh took: 0:24:06

Running Test...
  Accuracy: 0.448
  F1 Score: 0.398
  Test Loss: 2.721
  Test took: 0:00:08
```

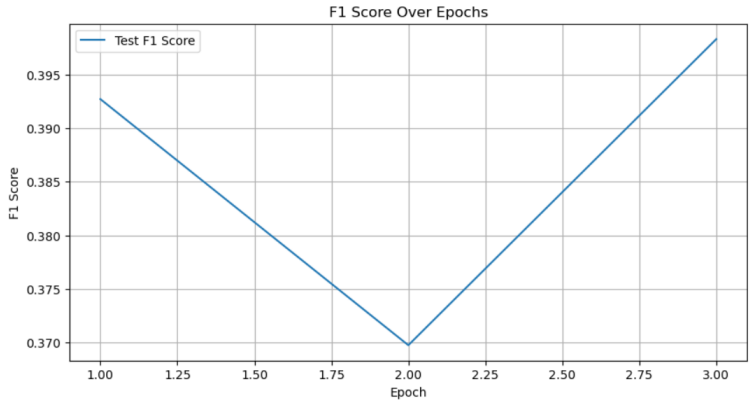
شکل ۴ـ۳: ایپوک ۳

```
{'epoch': 1, 'Training Loss generator': 0.6763042037461972, 'Training Loss discriminator': 1.9423223393367934, 'Valid. Loss': 1.8574364185333252, 'Valid. Accur.': 0.43666666666666665, 'Training Time': '0:18:25', 'Test Time': '0:00:08'}
{'epoch': 2, 'Training Loss generator': 0.7114604089084688, 'Training Loss discriminator': 1.1309138255434352, 'Valid. Loss': 2.202589988708496, 'Valid. Accur.': 0.4053333333333333, 'Training Time': '0:23:00', 'Test Time': '0:00:13'}
{'epoch': 3, 'Training Loss generator': 0.707479796431086, 'Training Loss discriminator': 0.8759377081920435, 'Valid. Loss': 2.7212116718292236, 'Valid. Accur.': 0.448, 'Training Time': '0:24:06', 'Test Time': '0:00:08'}

Training complete!
Total training took 1:06:01 (h:mm:ss)
```

شکل ۴ـ۴: خلاصه نتایج







- مدل G_1، ۴۰ درصد برچسب‌دار، ۶۰ درصد بدون برچسب:

با افزایش دیتاست برچسب دار نتیجه بهتر می شود، در مقایسه با حالت قبلی می توان دید که نوسان آموزش رفع شده و اگر حالت قبلی به ۱۰ ایپوک نیاز داشته باشد، این حالت به تعداد کمتری نیاز دارد.

```
  Average training loss generetor: 0.666
  Average training loss discriminator: 2.084
  Training epcoh took: 0:15:49

Running Test...
  Accuracy: 0.455
  F1 Score: 0.414
  Test Loss: 1.522
  Test took: 0:00:10
```

شکل ۴ـ۵: ایپوک ۱

```
  Average training loss generetor: 0.712
  Average training loss discriminator: 1.372
  Training epcoh took: 0:16:31

Running Test...
  Accuracy: 0.467
  F1 Score: 0.456
  Test Loss: 1.625
  Test took: 0:00:09
```

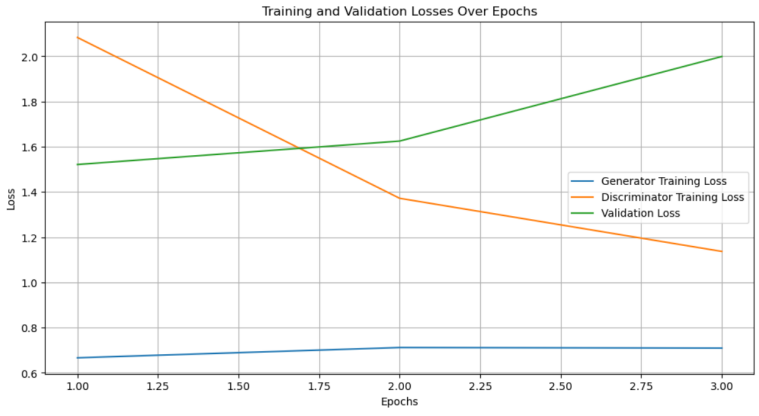
شکل ۴ـ۶: ایپوک ۲

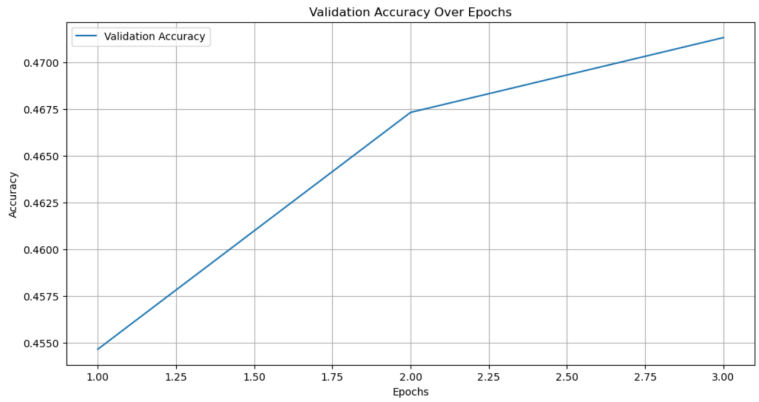
```
  Average training loss generetor: 0.709
  Average training loss discriminator: 1.137
  Training epcoh took: 0:17:34

Running Test...
  Accuracy: 0.471
  F1 Score: 0.440
  Test Loss: 1.999
  Test took: 0:00:13
```

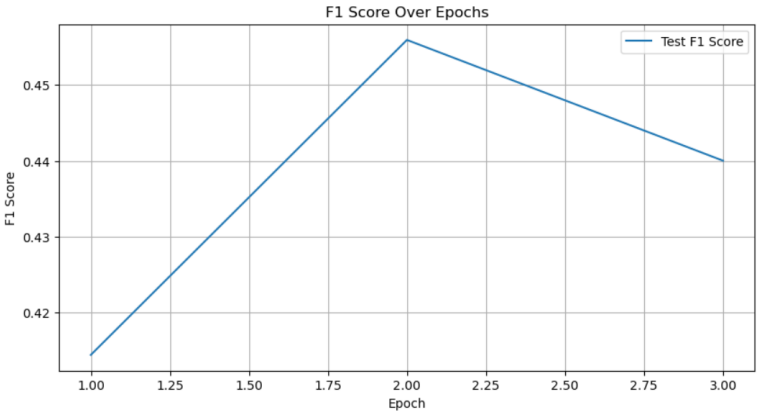
شکل ۴ـ۷: ایپوک ۳

```
{'epoch': 1, 'Training Loss generator': 0.6664814874932573, 'Training Loss discriminator': 2.0835476065541174, 'Valid. Loss': 1.5215482711791992, 'Valid. Accur.': 0.45466666666666666, 'Training Time': '0:15:49', 'Test Time': '0:00:10'}
{'epoch': 2, 'Training Loss generator': 0.7118143864580103, 'Training Loss discriminator': 1.372109925156241, 'Valid. Loss': 1.6251945495605469, 'Valid. Accur.': 0.4673333333333333, 'Training Time': '0:16:31', 'Test Time': '0:00:09'}
{'epoch': 3, 'Training Loss generator': 0.7094472833044894, 'Training Loss discriminator': 1.137035841823698, 'Valid. Loss': 1.999358057975769, 'Valid. Accur.': 0.4713333333333333, 'Training Time': '0:17:34', 'Test Time': '0:00:13'}

Training complete!
Total training took 0:50:26 (h:mm:ss)
```

شکل ۴ـ۸: خلاصه نتایج







تحلیل نتایج معماری G_1:

ـ Loss آموزش برای شبکه مولد:

$$Epoch1 : 0.6665$$

$$Epoch2 : 0.7118$$

$$Epoch3 : 0.7094$$

Loss شبکه مولد معیاری است که نشان می دهد چقدر قادر به تولید نمونه های واقعی است. مقادیر کمتر معمولاً بهتر هستند، زیرا نشان می دهد که شبکه مولد در ایجاد خروجی های واقعی در حال بهبود است. افزایش جزئی در تلفات شبکه مولد از ایپوک ۱ به ایپوک ۲ و سپس یک مقدار نسبتاً پایدار در ایپوک ۳ ممکن است نشان دهنده مقداری نوسان باشد، اما لزوماً نشان دهنده مشکل نیست.

ـ Loss آموزش برای شبکه تمییزدهنده:

$$Epoch1 : 2.0835$$

$$Epoch2 : 1.3721$$

$$Epoch3 : 1.1370$$

Loss تمییزدهنده میزان تمایز بین نمونه های واقعی و تولید شده را اندازه گیری می کند. مشابه Lossا شبکه مولد، مقادیر پایین تر به طور کلی بهتر است. کاهش قابل توجه در Loss تمایزدهنده از ایپوک ۱ تا ایپوک ۳ نشان می دهد که مولد در فریب دادن تمایزدهنده بهبود یافته است.

ـ Loss اعتبارسنجی و accuracy:

$$Epoch1 : \ Loss1.5215, \ Accuracy0.4547$$

$$Epoch2 : \ Loss1.6252, \ Accuracy0.4673$$

$$Epoch3 : \ Loss1.9994, \ Accuracy0.4713$$

Loss اعتبارسنجی نشان می دهد که چقدر مدل به داده های دیده نشده تعمیم می یابد. افزایش Loss اعتبار از ایپوک ۱ به ایپوک ۳ ممکن است بیش از حد را نشان دهد. دقت نسبتاً پایدار است اما خیلی بالا نیست.

ـ زمان آموزش و تست:

$$Epoch1 : \ 15minutes \ 49seconds$$

$$Epoch2 : \ 16minutes \ 31seconds$$

$$Epoch3 : \ 17minutes \ 34seconds$$

زمان آموزش با هر ایپوک در حال افزایش است، که می تواند به دلیل یادگیری الگوهای پیچیده تر مدل یا بزرگ بودن مجموعه داده باشد.

- مدل G_2، ۲۰ درصد برچسب‌دار، ۸۰ درصد بدون برچسب:

```
  Average training loss generetor: 0.722
  Average training loss discriminator: 1.655
  Training epcoh took: 0:19:29

Running Test...
  Accuracy: 0.441
  F1 Score: 0.398
  Test Loss: 1.918
  Test took: 0:00:08
```

شکل ۴ـ۹: ایپوک ۱

```
  Average training loss generetor: 0.712
  Average training loss discriminator: 1.051
  Training epcoh took: 0:26:17

Running Test...
  Accuracy: 0.436
  F1 Score: 0.387
  Test Loss: 2.312
  Test took: 0:00:13
```

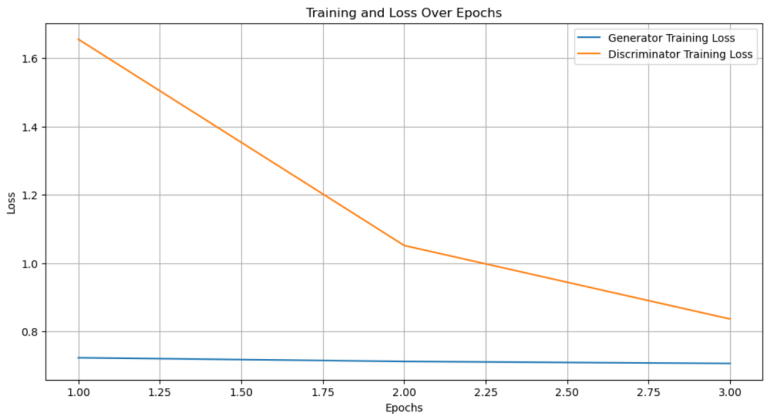
شکل ۴ـ۱۰: ایپوک ۲

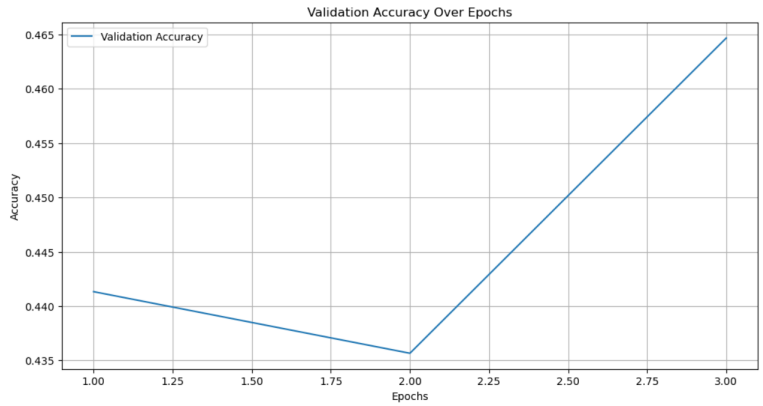
```
  Average training loss generetor: 0.706
  Average training loss discriminator: 0.836
  Training epcoh took: 0:22:20

Running Test...
  Accuracy: 0.465
  F1 Score: 0.426
  Test Loss: 2.879
  Test took: 0:00:11
```

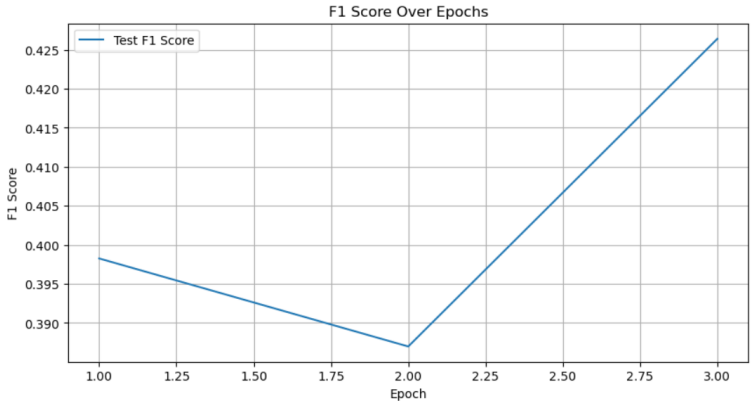
شکل ۴ـ۱۱: ایپوک ۳

```
{'epoch': 1, 'Training Loss generator': 0.7224929826529892, 'Training Loss discriminator': 1.6550412992009886, 'Valid. Accur.': 0.44133333333333336, 'Tra
ining Time': '0:19:29', 'Test Time': '0:00:08'}
{'epoch': 2, 'Training Loss generator': 0.7116552988240669, 'Training Loss discriminator': 1.0511942154115386, 'Valid. Accur.': 0.43566666666666665, 'Tra
ining Time': '0:26:17', 'Test Time': '0:00:13'}
{'epoch': 3, 'Training Loss generator': 0.7057131770942304, 'Training Loss discriminator': 0.8359519980243735, 'Valid. Accur.': 0.4646666666666667, 'Trai
ning Time': '0:22:20', 'Test Time': '0:00:11'}

Training complete!
Total training took 1:08:37 (h:mm:ss)
```

شکل ۴ـ۱۲: خلاصه نتایج







دقت در G_1 برابر 0.436 شد و در G_2 0.465 شد.

- مدل G_2، ۵۰ درصد برچسب‌دار، ۵۰ درصد بدون برچسب:

  با افزایش دیتاست برچسب دار نتیجه بهتر می شود:

```
  Average training loss genereтor: 0.731
  Average training loss discriminator: 1.754
  Training epcoh took: 0:22:24

Running Test...
  Accuracy: 0.464
  F1 Score: 0.414
  Test Loss: 1.524
  Test took: 0:00:07
```

شکل ۴ـ۱۳: ایپوک ۱

```
  Average training loss genereтor: 0.720
  Average training loss discriminator: 1.276
  Training epcoh took: 0:16:59

Running Test...
  Accuracy: 0.506
  F1 Score: 0.487
  Test Loss: 1.689
  Test took: 0:00:11
```

شکل ۴ـ۱۴: ایپوک ۲

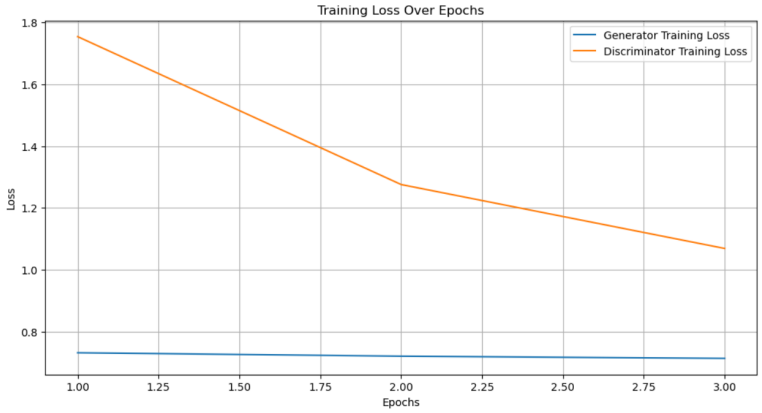
```
  Average training loss genereтor: 0.713
  Average training loss discriminator: 1.069
  Training epcoh took: 0:17:27

Running Test...
  Accuracy: 0.501
  F1 Score: 0.472
  Test Loss: 2.024
  Test took: 0:00:10
```

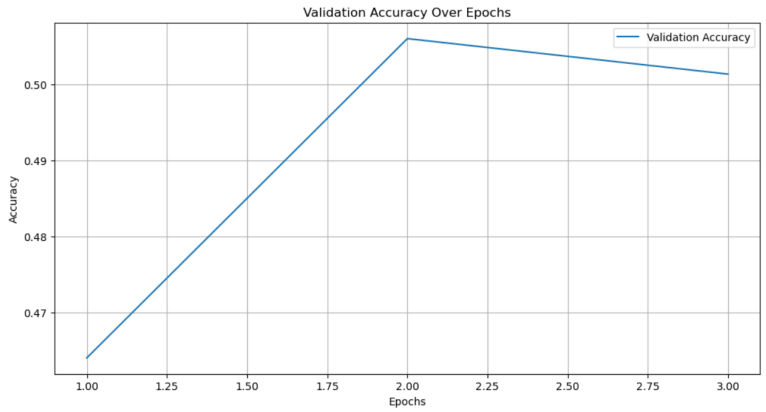
شکل ۴ـ۱۵: ایپوک ۳

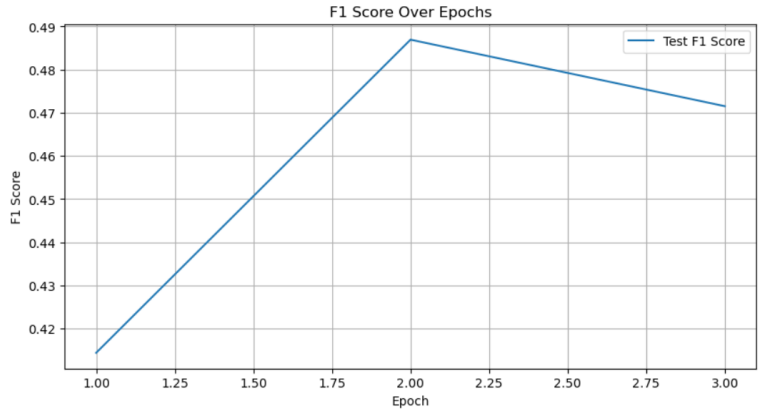
```
{'epoch': 1, 'Training Loss generator': 0.7310682469391608, 'Training Loss discriminator': 1.7543531511280988, 'Valid. Accur.': 0.464, 'Training Time':
'0:22:24', 'Test Time': '0:00:07'}
{'epoch': 2, 'Training Loss generator': 0.7199606829935365, 'Training Loss discriminator': 1.2755506262048946, 'Valid. Accur.': 0.506, 'Training Time':
'0:16:59', 'Test Time': '0:00:11'}
{'epoch': 3, 'Training Loss generator': 0.7128319162237752, 'Training Loss discriminator': 1.0686820583300547, 'Valid. Accur.': 0.5013333333333333, 'Trai
ning Time': '0:17:27', 'Test Time': '0:00:10'}

Training complete!
Total training took 0:57:17 (h:mm:ss)
```

شکل ۴ـ۱۶: خلاصه نتایج

دقت در G_1 0.51 شد(۴ ایپوکـ۱ ساعت و ۱۴ دقیقه) و در G_2 0.501 شد منتها با (۳ایپوکـ۵۷ دقیقه).







همانطور که مشاهده شد در مجموع نتایج برای G_2 بهتر از G_1 است که دلایل آن در زیربخش قبل تشریح شد.

فصل ۵

خواسته چهارم (امتیازی)

در این فصل به بهبود GAN-BERT با معماری G_1 می‌پردازیم.

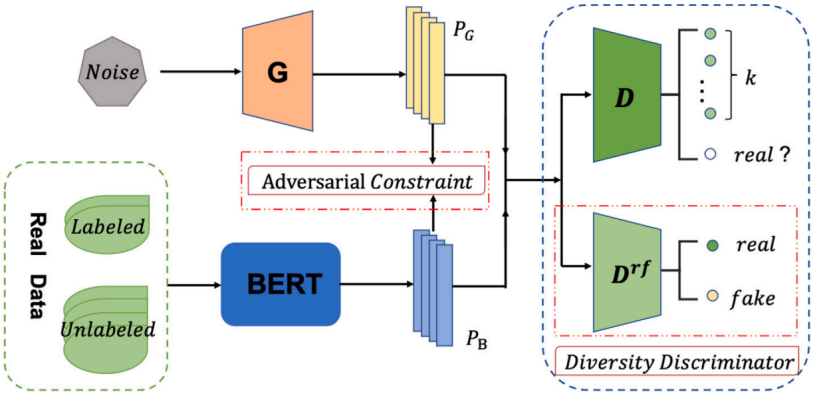
۵ـ۱ تشریح روش

برای بهبود مدل دو دسته اقدامات صورت گرفته است:

۱. بهبود شبکه مولد با استفاده از تغییرات زیر:

- تنظیم مقدار dropout
- افزودن لایه Batch Normalization به لایه‌های مخفی

۲. استفاده از دو شبکه مجزای تمییزدهنده، یکی برای تشخیص واقعی یا جعلی بودن نمونه‌ها و دیگری به عنوان طبقه‌بند (شکل ۵ـ۱)



شکل ۵ـ۱: GAN-BERT با دو discriminator

۵ـ۲ نتایج

مراجع

[1] Kenton, Jacob Devlin Ming-Wei Chang and Toutanova, Lee Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. in *Proceedings of naacL-HLT*, vol. 1, p. 2, 2019.

[2] Croce, Danilo, Castellucci, Giuseppe, and Basili, Roberto. Gan-bert: Generative adversarial learning for robust text classification with a bunch of labeled examples. 2020.