تمرین کامپیوتری اول – Tokenization

ثمین مهدی زاده – ۸۱۰۱۰۰۵۲۶

در این تمرین به بررسی روش های Tokenization پرداخته شده است. در ابتدا سعی شده که دو روش BPE و wordPiece معرفی شود و سپس به مقایسه آن ها پرداخته می شود. در این تمرین همچنین روش BPE پیاده سازی شده است.

گام اول

BPE

در این روش در ابتدا داده های آموزش به کمک یک pre-tokenizer به تعدادی کلمه شکسته می شوند.pre-tokenizer ها می توانند مدل های ساده ای باشند که تنها داده ها را بر اساس space می شکنند و یا مدل های پیچیده تری باشند که بر اساس تعدادی قانون کار می کنند. پس از شکستن متنُ، کلمات مجزای به وجود آمده و فرکانس تکرار هر کدام از آن ها در متن به دست می آید. سپس یک vacbulary پایه که متشکل از تمام حروف موجود در کلمات منحصر به فرد است به وجود می آید و در هر مرحله هر کدام از نماد های موجود در vocabulary(که می توانند یک یا چند حرف باشند) به صورت دو به دو با یکدیگر ترکیب می شوند و هر کدام که تعداد تکرار بیشتری داشته باشد به عنوان نماد جدید وارد vocabulary می شود. این کار تا زمانی ادامه پیدا می کند نماد های موجود در vocabulary به تعداد مورد نظر برسد.

wordPiece

یکی دیگر از روش های tokenization الگوریتم wordPriece است. این الگوریتم شباهت زیادی به BPE دارد. در این روش نیز پس از به وجود آمدن vocabulary (که در ابتدا تنها شامل حروف منحصر به فرد موجود در متن هست) در هر مرحله هر یک از نماد ها با یکدیگر ادغام شده و نماد جدیدی را وارد vocabulary می کند. اما تفاوتی که این الگوریتم با BPE دارد این است که در این جا برای انتخاب بهترین جفت از پر تکرار ترین آن ها استفاده نمی شود بلکه جفتی انتخاب می شود که احتمال یا likelihood داده های آموزش را بیشینه می کند.

در این گام الگوریتم BPE پیاده سازی شد. در ابتدا corpus زیر به عنوان داده ی آموزشی به مدل داده شد و سپس چگونگی tokenize کردن کلمه lower توسط این روش به دست آمد. لازم به ذکر است که در این جا برای جداسازی کلمات تنها بر اساس space بوده است.

متن آموزشی:

```
corpus = "low low low low low lower lower widest widest widest newest newest newest newest newest"
```

مراحل طی شده برای ساخت vocab:

```
------------ iteration 0 ------------

pairs: {('l', 'o'): 7, ('o', 'w'): 7, ('w', 'e'): 7, ('e', 'r'): 2, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'e'): 3, ('e', 's'): 8, ('s', 't'): 8, ('n', 'e'): 5, ('e', 'w'): 5}
best pair:  es
dictionay: {'l o w': 5, 'l o w e r': 2, 'w i d es t': 3, 'n e w es t': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es']


------------ iteration 1 ------------

pairs: {('l', 'o'): 7, ('o', 'w'): 7, ('w', 'e'): 2, ('e', 'r'): 2, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'es'): 3, ('es', 't'): 8, ('n', 'e'): 5, ('e', 'w'): 5, ('w', 'es'): 5}
best pair:  est
dictionay: {'l o w': 5, 'l o w e r': 2, 'w i d est': 3, 'n e w est': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est']


------------ iteration 2 ------------

pairs: {('l', 'o'): 7, ('o', 'w'): 7, ('w', 'e'): 2, ('e', 'r'): 2, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est'): 3, ('n', 'e'): 5, ('e', 'w'): 5, ('w', 'est'): 5}
best pair:  lo
dictionay: {'lo w': 5, 'lo w e r': 2, 'w i d est': 3, 'n e w est': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est', 'lo']
```

```
---------- iteration 3 ----------

pairs: {('lo', 'w'): 7, ('w', 'e'): 2, ('e', 'r'): 2, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est'): 3, ('n', 'e'): 5, ('e', 'w'): 5, ('w', 'est'): 5}
best pair:  low
dictionay: {'low': 5, 'low e r': 2, 'w i d est': 3, 'n e w est': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est', 'lo', 'low']

---------- iteration 4 ----------

pairs: {('low', 'e'): 2, ('e', 'r'): 2, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est'): 3, ('n', 'e'): 5, ('e', 'w'): 5, ('w', 'est'): 5}
best pair:  ne
dictionay: {'low': 5, 'low e r': 2, 'w i d est': 3, 'ne w est': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est', 'lo', 'low', 'ne']

---------- iteration 5 ----------

pairs: {('low', 'e'): 2, ('e', 'r'): 2, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est'): 3, ('ne', 'w'): 5, ('w', 'est'): 5}
best pair:  new
dictionay: {'low': 5, 'low e r': 2, 'w i d est': 3, 'new est': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est', 'lo', 'low', 'ne', 'new']

---------- iteration 6 ----------

pairs: {('low', 'e'): 2, ('e', 'r'): 2, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est'): 3, ('new', 'est'): 5}
best pair:  newest
dictionay: {'low': 5, 'low e r': 2, 'w i d est': 3, 'newest': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est', 'lo', 'low', 'ne', 'new', 'newest']

---------- iteration 7 ----------

pairs: {('low', 'e'): 2, ('e', 'r'): 2, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est'): 3}
best pair:  wi
dictionay: {'low': 5, 'low e r': 2, 'wi d est': 3, 'newest': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est', 'lo', 'low', 'ne', 'new', 'newest', 'wi']

---------- iteration 8 ----------

pairs: {('low', 'e'): 2, ('e', 'r'): 2, ('wi', 'd'): 3, ('d', 'est'): 3}
best pair:  wid
dictionay: {'low': 5, 'low e r': 2, 'wid est': 3, 'newest': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est', 'lo', 'low', 'ne', 'new', 'newest', 'wi', 'wid']

---------- iteration 9 ----------

pairs: {('low', 'e'): 2, ('e', 'r'): 2, ('wid', 'est'): 3}
best pair:  widest
dictionay: {'low': 5, 'low e r': 2, 'widest': 3, 'newest': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est', 'lo', 'low', 'ne', 'new', 'newest', 'wi', 'wid', 'widest']

---------- iteration 10 ----------

pairs: {('low', 'e'): 2, ('e', 'r'): 2}
best pair:  lowe
dictionay: {'low': 5, 'lowe r': 2, 'widest': 3, 'newest': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est', 'lo', 'low', 'ne', 'new', 'newest', 'wi', 'wid', 'widest', 'lowe']

---------- iteration 11 ----------

pairs: {('lowe', 'r'): 2}
best pair:  lower
dictionay: {'low': 5, 'lower': 2, 'widest': 3, 'newest': 5}
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est', 'lo', 'low', 'ne', 'new', 'newest', 'wi', 'wid', 'widest', 'lowe', 'lower']
```

همان طور که تصاویر بالا نیز نشان می دهد در هر مرحله جفت های دو تایی از نماد ها به وجود آمده و جفت پر تکرار به عنوان نماد بعدی وارد vocabulary می شود. برای مثال مشاهده می شود که در iteration 0 هر کدام از حروف موجود در corpus به صورت دو به دو ترکیب شده و جفتی که بیشترین تکرار را داشته یعنی es وارد vocabulary شده و سپس تمام کلماتی که در آن ها دو حرف e و s به صورت پشت سر هم وجود داشت تغییر کرده و این دو حرف به صورت چسبیده به هم قرار گرفتند.(l o w e s t به l o w es t) تبدیل شده است.

در انتها نیز مشاهده می شود که پس از ۱۲ بار تکرار vocab استخراج شده به شکل زیر است:

```
vocab: ['d', 't', 'n', 'e', 'i', 'l', 's', 'w', 'o', 'r', 'es', 'est', 'lo', 'low', 'ne', 'new', 'newest', 'wi', 'wid', 'widest', 'lowe', 'lower']
```

جهت tokenize کردن کلمه ی lower لازم است ابتدا تمام کاراکتر های موجود در آن جدا شود و سپس این کاراکتر ها به صورت دو به دو با یکدیگر ترکیب می شود و هر ترکیبی که در vocabulary وجود داشت انتخاب شده و به صورت چسبیده به هم در کلمه قرار می گیرد. در صورتی که چندین جفت در vocabulary وجود داشت جفتی انتخاب می شود که زود تر در vocabulary ظاهر شده است(برای مثال میان دو جفت es و lo به es اولویت داده می شود).

در ادامه پس از تولید vocabulary به کمک داده های آموزش، tokenization بر روی کلمه ی lowest صورت می گیرد:

```
bpe = BPE_Tokenizer(12)
bpe.generate_vocab(corpus)
bpe.tokenize("lowest")
```

```
---------- tokenizing lowest ----------

(es) - merged word:  ['l', 'o', 'w', 'es', 't']
(est) - merged word:  ['l', 'o', 'w', 'est']
(lo) - merged word:  ['lo', 'w', 'est']
(low) - merged word:  ['low', 'est']
```

همان طور که مشاهده می شود در این جا با این که خود کلمه در corpus وجود نداشت اما به دلیل توجه الگوریتم BPE به subword ها و اضافه کردن آن ها به vocabulary کلمه lowest به دو جز low و est شکسته شد.

گام دوم

در این گام به کمک کتابخانه hugging face دو الگوریتم گفته شده بر روی داده های ویکی پدیا و کتاب کوتنبرگ مورد بررسی قرار گرفته اند و سپس متن زیر برای tokenization به آنها داده شد. همچنین در انتها اطلاعات مربوط به هر کدام از این tokenizer ها در فایل هایی ذخیره شده اند تا در گام بعدی از آن استفاده شود.

```
This is a deep learning tokenization tutorial. Tokenization is the first step in a
deep learning NLP pipeline. We will be comparing the tokens generated by each
tokenization model. Excited much?! 🤩
```

نتایج به دست آمده از اجرای این الگوریتم ها به شکل زیر است:

```
-------------- wp-tokenizer-guten --------------


tokenize: ['This', 'is', 'a', 'deep', 'learning', 'to', '##ken', '##ization', 't', '##ut', '##oria', '##l', '.', 'To',
'##ken', '##ization', 'is', 'the', 'first', 'step', 'in', 'a', 'deep', 'learning', 'N', '##L', '##P', 'pip', '##el',
'##ine', '.', 'We', 'will', 'be', 'comparing', 'the', 'to', '##ken', '##s', 'generated', 'by', 'each', 'to', '##ken',
'##ization', 'model', '.', 'Ex', '##ci', '##ted', 'much', '[UNK]']
size: 52


-------------- wp-tokenizer-wiki --------------


tokenize: ['This', 'is', 'a', 'deep', 'learning', 'to', '##ken', '##ization', 'tut', '##orial', '.', 'Tok', '##eni',
'##za', '##ti', '##on', 'is', 'the', 'first', 'step', 'in', 'a', 'deep', 'learning', 'NL', '##P', 'pipeline', '.',
'We', 'will', 'be', 'comparing', 'the', 'to', '##ken', '##s', 'generated', 'by', 'each', 'to', '##ken', '##ization',
'model', '.', 'Exc', '##ited', 'much', '[UNK]']
size: 48


-------------- bpe-tokenizer-guten --------------


tokenize: ['This', 'is', 'a', 'deep', 'learning', 'to', 'ken', 'ization', 't', 'ut', 'or', 'ial', '.', 'T', 'ok', 'en',
'ization', 'is', 'the', 'first', 'step', 'in', 'a', 'deep', 'learning', 'N', 'L', 'P', 'pi', 'pe', 'line', '.', 'We',
'will', 'be', 'comparing', 'the', 'to', 'k', 'ens', 'generated', 'by', 'each', 'to', 'ken', 'ization', 'model', '.',
'Ex', 'c', 'ited', 'much', '?', '!', '[UNK]']
size: 55


-------------- bpe-tokenizer-wiki --------------


tokenize: ['This', 'is', 'a', 'deep', 'learning', 'to', 'ken', 'ization', 'tut', 'orial', '.', 'Tok', 'en', 'ization',
'is', 'the', 'first', 'step', 'in', 'a', 'deep', 'learning', 'NL', 'P', 'pipeline', '.', 'We', 'will', 'be',
'comparing', 'the', 'tok', 'ens', 'generated', 'by', 'each', 'to', 'ken', 'ization', 'model', '.', 'Ex', 'cited',
'much', '?', '!', '[UNK]']
size: 47
```

یکی از بارز ترین تفاوت هایی که متوجه آن می شویم این است که در الگوریتم wordpiece قبل از برخی توکن ها علامت # وجود دارد. این موضوع می تواند هنگام tokenize کردن یک کلمه جدید به ما کمک کند. به این صورت که می دانیم اگر قرار است کلمه ای به توکن مورد نظر شکسته شود حتما باید قبل از آن نماد دیگری وجود داشته باشد و به آن چسبیده باشد. جدول زیر نمایانگر تفاوت های موجود میان دو الگوریتم در داده های آموزشی است:

کتاب گوتنبرگ		
wordPiece	BPE	کلمه
t,ut,oria,l	t,ut,or,ial	tutorial
To,ken,ization	T,ok,en,ization	Tokenization
pip,el,line	pi,pe,line	pipeline
to,ken,s	to,k,ens	tokens
Ex,ci,ted	Ex,c,ited	Excited

ویکی پدیا		
wordPiece	BPE	کلمه
Tok,eni,za,ti,on	Tok,en,ization	Tokenization
Exc,ited	Ex,cited	Excited

یکی دیگر از تفاوت هایی که می توان میان BPE و wordpiece دید این است که ؟! در توکن های wordpiece وجود ندارد و احتمالا جز UNK به شمار رفته باشد در حالی که در الگوریتم BPE به ازای هر یک توکن مجزا داریم.
نکته دیگری که در این جا باید به آن اشاره کرد این است که شکست کلمه ی tokenization و Tokenization در همه ی حالات به جز حالت اول(الگوریتم wordPiece با داده آموزشی گوتنبرگ) متفاوت از یکدیگر بوده است.

برای مقایسه تاثیر اندازه داده های آموزشی نیز می توان جداول زیر را در نظر گرفت که بیانگر تفاوت شکست کلمه بر اساس داده های مختلف است:

BPE		
کلمه	گوتنبرگ	ویکی پدیا
tutorial	t,ut,or,ial	tut,orial
Tokenization	T,ok,en,ization	Tok,en,ization
NLP	N,L,P	NL,P
pipeline	pi,pe,line	pipeline
tokens	to,k,ens	tok,ens
Excited	Ex,c,ited	Ex,cited

wordPiece		
کلمه	گوتنبرگ	ویکی پدیا
tutorial	t,ut,oria,l	tut,orial
Tokenization	To,ken,ization	Tok,eni,za,ti,on
NLP	N,L,P	NL,P
pipeline	pip,el,ine	pipeline
Excited	Ex,ci,ted	Exc,ited

جداول بالا نشان می دهد که هنگامی که از داده های ویکی پدیا استفاده می شود تعداد شکست ها تقریبا کمتر از حالتی است که الگوریتم wordPiece استفاده می کنیم و تقریبا کلمات بهتر شده اند برای مثال کلمه pipeline که در داده های ویکی پدیا یک توکن مجزا در نظر گرفته شده در داده های گوتنبرک به سه جز جدا از هم شکسته شده است.
به طور کلی نیز می توان گفت هرچه تعداد داده های آموزش بیشتر باشد مدل نمونه های بیشتری می بیند و در نتیجه احتمالا نتایج بهتری بر روی داده های دیده نشده خواهد داشت.(در صورتی که مدل بر روی داده های آموزش overfit نشود)
در این جا نیز و با توجه به جداول می توان دید هنگامی که از داده های ویکی استفاده شده تفاوت دو الگوریتم در tokenize کردن متن ورودی کمتر شده و نتایج بیشتر به یکدیگر نزدیک هستند.

گام سوم

در این گام متن کتاب گوتنبرگ به هر یک از مدل های گام قبل داده شده و تعداد توکن های هریک به دست آمده است:

```
wp-guten : 122738
wp-wiki : 140734
bpe-guten : 122738
bpe-wiki : 140871
```

ردیف	نام الگوریتم استفاده شده برای توکنایزر	تعداد توکن های خروجی الگوریتم برای کتاب گوتنبرگ	
		توکنایزر آموزش داده شده بر روی کتاب گوتنبرگ	توکنایزر آموزش داده شده بر روی کل داده های ویکی پدیا
۱	Byte Pair Encoding	۱۲۲۷۳۸	۱۴۰۸۷۱
۲	WordPiece	۱۲۲۷۳۸	۱۴۰۷۳۴

در این گام نیز می توان دید نتیجه توکنایز برای هر دو الگوریتم BPE و wordPiece هنگامی که داده آموزشی گوتنبرگ بوده یکسان است. در واقع می توان گفت هنگامی که یک توکنایزر بر روی داده هایی که توسط آن آموزش داده شده است تست می شود از آن جا که تمام متن ورودی را قبلا دیده است می تواند کلمات را به طور کامل توکنایز کند و به نتیجه بهتری برسد اما این موضوع نمی تواند بیانگر این باشد که بر روی سایر داده های آموزشی نیز بهتر عمل می کند. در اینجا علت بهتر عمل کردن توکنایزر گوتنبرگ به علت مشاهده داده های تست بوده و احتمالا توکنایزر ویکی پدیا به دلیل آموزش بر روی ورودی های بیشتر در سایر موادر بهتر عمل خواهد کرد. هم چنین می توان دید که الگوریتم wordPiece(که به کمک ویکی پدیا آموزش داده شده است) به تعداد توکن های کمتری رسیده است و نتیجه آن کمی به نتیجه به دست آمده بر روی مدل های آموزش دیده شده توسط گوتنبرگ نزدیک تر است بنابراین می توان احتمالا این الگوریتم نسب به BPE بهتر عمل کرده است.

❋ برای اجرای کد لازم است تا داده های آموزش درون فولدر data که درون همان دایرکتوری کد است و با نام های زیر قرار بگیرند.