به نام خدا

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

پردازش زبان‌های طبیعی

تمرین اول

نام و نام خانوادگی : حسین سیفی

شماره دانشجویی : ۸۱۰۱۰۰۳۸۶

اسفند ۱۴۰۰

فهرست

مقدمه

در این تمرین به بررسی و مقایسه دو روش Byte Pair Encoding Tokenization و WordPiece Tokenization می‌پردازیم.

گام اول

روش کار دو الگوریتم ذکر شده به شکل زیر می‌باشد:

- Byte Pair Encoding Tokenization: در این روش ابتدا متن آموزشی با استفاده از فواصل بین کلمات به مجموعه‌ای از کلمات تقسیم می‌شود و کلمات تکراری حذف شده و تعداد تکرار هر کلمه در کنار آن قید می‌شود در نتیجه مجموعه‌ای از کلمات یکتا که به صورت مجموعه‌ای از حروف متوالی و مجزا است را همراه با فرکانس ظهور آنها در داده‌های آموزشی در دسترس است. سپس BPE یک لغت‌نامه (Vocabulary) شامل تمام حروف(نمادها) موجود در کلمات یکتا ایجاد می‌کند. در مرحله بعد BPE فرکانس هر زوج از نمادهای ممکن را محاسبه می‌کند و زوج نمادهایی با بیشترین فرکانس را انتخاب کرده و این زوج را به لغت‌نامه اضافه می‌کند و در تمامی کلماتی که شامل این توالی هستند ، نمادها را با یکدیگر ترکیب می‌کند. این محاسبه فرکانس و ترکیب نمادها تا جایی ادامه می‌یابد که به نتیجه مطلوب منجر شود. نتیجه مطلوب می‌تواند رسیدن به اندازه مشخص لغت‌نامه، رسیدن به تعداد مشخصی ترکیب یا ایجاد تمامی کلمات ممکن باشد.
- WordPiece Tokenization: در این روش نیز مانند BPE ، لغت‌نامه‌ای از تمام حروف موجود در متن آموزشی ایجاد می‌کنیم ، سپس از بین تمام زوج نمادهای ممکن ، زوجی را انتخاب می‌کند که در صورت اضافه شدن به لغت‌نامه مقدار Likelihood دادگان آموزشی را به حداکثر مقدار خود برساند. برای انجام این کار زوج "XY" را انتخاب می‌کنیم که بیشترین مقدار $\frac{P("XY")}{P("X")P("Y")}$ را تولید کند. این الگوریتم زمانی به نتیجه نهایی خود می‌رسد که با توجه به لغت‌نامه هیچ زوج دیگری از اعضای متن آموزشی امکان به هم پیوستن را نداشته باشند یا به تعداد اعضای مورد در در لغت‌نامه برسیم.

شباهت‌ها:

۱. هر دو الگوریتم به دنبال زوجی از کلمات(نمادها) لغت‌نامه هستند که دارای ویژگی خاصی باشند و با ترکیب آنها، اعضای لغت‌نامه با معنی‌تر باشند و تشکیل کلماتی واقعی را بدهند.

۲. هر دو الگوریتم در شرایط تقریبا مشابهی به پایان می‌رسند و نیاز به شرطی از جانب طراح مدل نیاز دارند ، در غیر اینصورت هنگامی که تمامی کلمات ممکن را به دیکشنری اضافه کنند و امکان ترکیب زوجی دیگر وجود نداشته باشد.

تفاوت‌ها:

۱. انتخاب زوج نماد مورد نظر برای ترکیب و الصاق در دو الگوریتم به روش متفاوتی انجام می‌گیرد. الگوریتم BPE صرفا بر اساس فرکانس پیدایش زوج نمادها در متن آموزش آن‌ها را ترکیب می‌کند اما الگوریتم WordPiece تلاش می‌کند که مقدار Likelihood را به بیشترین مقدار خود برساند. این کار با محاسبه $\frac{P("XY")}{P("X")P("Y")}$ برای رشته‌ی "XY" انجام می‌گیرد.

۲. الگوریتم BPE شامل مرحله پیش-توکن‌سازی نیز می‌شود که در این مرحله باید متن آموزش از فاصله بین کلمات شکسته شود اما WordPiece شامل چنین مرحله‌ای نیست. با توجه به اینکه الگوریتم WordPiece در زبان‌های چینی و ژاپنی

که فاصله بین کلمات ندارند مورد استفاده قرار می‌گیرد و الگوریتم BPE در زبان‌هایی مانند انگلیسی استفاده می‌شود، این تفاوت قابل توجیه است.

پیاده سازی BPE:

الگوریتم BPE به کمک زبان برنامه نویسی پایتون پیاده سازی شده است. این الگوریتم به روش برنامه نویسی پویا پیاده سازی شده است و جدولی به منظور نگهداری فرکانس هر زوج نماد لغت‌نامه ایجاد می‌شود. در هر تکرار فقط مقدار خانه‌هایی از جدول تغییر می‌کنند که در ترکیب نمادهای انجام شده‌ی اخیر ، دخالت داشته‌اند. جزئیات کد نوشته شده در ادامه توضیح داده خواهد شد:

کتابخانه‌ها:

در این پیاده سازی از کتابخانه Numpy استفاده شده است.

ایجاد لغت‌نامه:

در قطعه کد زیر حروف موجود در داده‌های ورودی الگوریتم استخراج شده و در یک لیست نگه‌داری می‌شوند. همچنین کاراکتر "_" به عنوان نشانه‌ای برای پایان کلمات مورد استفاده قرار می‌گیرد و به لغت‌نامه اضافه می‌شود.

```
def CreateVocab(corpus):
    v = list(set(corpus)) +['_']
    v.remove(' ')
    return v
```

آماده سازی داده ورودی:

داده‌های ورودی از فاصله بین کلمات شکسته می‌شود و به عنوان یک لیست از حروف ذخیره می‌شود. کاراکتر "_" نیز به عنوان کاراکتر پایانی به کلمات اضافه می‌شود.

```
def CleanCorpus(corpus):
    corpus = corpus.split(' ')
    for i in range(len(corpus)):
        corpus[i] = list(corpus[i]) + ['_']
    return corpus
```

محاسبه فرکانس زوج نمادها:

تابع زیر فرکانس زوج نماد مورد نظر را در داده‌های ورودی محاسبه می‌کند.

```
def CalcFreq(corpus, str1, str2):
    f = 0
    for w in corpus:
        for l in range(len(w)-1):
            if w[l] == str1 and w[l+1] == str2:
                f += 1
    return f
```

مقداردهی اولیه ماتریس فرکانس :

ماتریس مورد استفاده برای ذخیره فرکانس زوج نمادها با استفاده از شکل اولیه داده‌های آموزشی به شکل زیر مقداردهی اولیه می‌شود.

```
def InitialSetupMatrix(corpus, FreqMatrix, v):

    for i in range(len(v)):
        for j in range(len(v)):
            FreqMatrix[i,j] = CalcFreq(corpus, v[i], v[j])
```

ترکیب زوج نمادها:

این تابع برای حذف دو نماد یا حرف مورد نظر و جایگزینی آنها با ترکیب آن دو حرف در دادگان آموزشی استفاده دارد.

```
def MergeLetters(corpus, str1, str2):

    for w in corpus:
        for l in range(len(w)-1,0,-1):
            if w[l-1] == str1 and w[l] == str2:
                del w[l]
                del w[l]
                w.insert(l, (str1+str2))
```

گسترش ماتریس فرکانس:

```
def ExtendMatrix(FreqMatrix, v, str1, str2, corpus):

    NewStr = str1+str2
    if NewStr in v:
        return FreqMatrix
    print(NewStr)
    MergeLetters(corpus, str1, str2)
    v.append(NewStr)
    n = FreqMatrix.shape[0]
    temp = np.full((n+1,n+1),-1)
    temp[:n,:n] = FreqMatrix
    s1Index = v.index(str1)
    s2Index = v.index(str2)
    for i in range(n+1):
        temp[i,n] = CalcFreq(corpus, v[i], NewStr)
        temp[n,i] = CalcFreq(corpus, NewStr, v[i])
        temp[i,s1Index] = CalcFreq(corpus, v[i], str1)
        temp[s1Index,i] = CalcFreq(corpus, str1, v[i])
        temp[i,s2Index] = CalcFreq(corpus, v[i], str2)
        temp[s2Index,i] = CalcFreq(corpus, str2, v[i])
    return temp
```

تابع فوق وظیفه اضافه کردن نماد جدید که ترکیبی از دو نماد پرتکرار قبلی است را به لغت‌نامه بر عهده دارد. همچنین سطر و ستونی جدید برای این نماد در ماتریس فرکانس تعریف می‌کند و مقادیر نمادهای ترکیب شده در ماتریس را آپدیت می‌کند.

تابع نهایی:

با فراخوانی تابع زیر الگوریتم BPE روی داده‌های مورد نظر اجرا می‌شود. روند اجرای این تابع مانند الگوریتم BPE بدین صورت است که ابتدا لغت‌نامه‌ای با استفاده از دادگان آموزشی ایجاد و دادگان آموزشی به فرمت مورد نظر تبدیل می‌شود. با توجه به اندازه فعلی لغت‌نامه ماتریس فرکانس نیز ایجاد و مقدار دهی اولیه می‌شود. سپس تا زمانی که یک زوج نماد برای ترکیب وجود داشته باشد و فرکانسی بیشتر از صفر داشته باشد ، اندیس آن را پیدا می‌کنیم و عمل گسترش ماتریس و ترکیب نمادها را انجام می‌دهیم.

```
def BytePairEncoding(Input_text):

    v = CreateVocab(Input_text)
    corpus = CleanCorpus(Input_text)
    FreqMatrix = np.full((len(v),len(v)),-1)
    InitialSetupMatrix(corpus, FreqMatrix, v)
    MaxIndex = np.unravel_index(FreqMatrix.argmax(), FreqMatrix.shape)
    while not np.max(FreqMatrix) == 0:
        temp = ExtendMatrix(FreqMatrix, v, v[MaxIndex[0]], v[MaxIndex[1]], corpus)
        MaxIndex = np.unravel_index(FreqMatrix.argmax(), FreqMatrix.shape)
        del FreqMatrix
        FreqMatrix = temp
    return v
```

مثالی برای اجرای الگوریتم:

در این بخش نحوه اجرای مرحله به مرحله الگوریتم BPE را بر روی مثال داده شده بررسی می‌کنیم. پس از انجام مرحله اول پردازش(ایجاد لغت‌نامه، شکستن داده‌ها از فواصل بین کلمات و تبدیل کلمات به لیستی از حروف)، اطلاعات مورد نیاز به شکل زیر درمی‌آیند:

5 L o w _
2 L o w e r _
3 W i d e s t _
5 N e w e s t _

V = {d,e,i,l,n,o,r,s,t,w,_}

سه ترکیب ابتدایی زوج حروف (“t”,”_”) و (“s”,”t_”) و (“e”,”st_”) هر کدام با فرکانس ۸ می‌باشند. هر ترتیبی که منجر به ایجاد رشته “est_” شود نیز ممکن است توسط این الگوریتم انجام شود. پس از این مرحله دادگان به شرح زیر می‌باشند:

5 L o w _
2 L o w e r _
3 W i d est_
5 N e w est_

V = {d,e,i,l,n,o,r,s,t,w,_,t_,st_,est_}

در مرحله بعد با هر ترتیب ممکن که رشته “low” ایجاد شود، زوج‌های (“lo”,”w”) و (“l”,”o”) هر کدام با فرکانس ۷ ترکیب می‌شوند. در نتیجه دادگان به شکل زیر درمی‌آیند:

5 Low _
2 Low e r _ V = {d,e,i,l,n,o,r,s,t,w,_,t_,st_,est_,lo,low}
3 W i d est_
5 N e w est_

در گام بعدی زوج حروف (“e”,”w”) و سپس (“n”,”ew”) هر کدام با فرکانس ۵ منجر به ایجاد رشته ”new“ می‌شوند. همچنین زوج حروف (”low”,”_”) نیز با فرکانس مشابه ۵ قابل ترکیب هستند. شکل جدید دادگان در نمایش زیر قابل مشاهده است:

5 Low_
2 Low e r _ V = {d,e,i,l,n,o,r,s,t,w,_,t_,st_,est_,lo,low,ew,new,low_}
3 W i d est_
5 New est_

پس از انجام ترکیب‌های مرحله قبل، امکان ترکیب زوج حرف (“new”,”est”) با فرکانس ۵ وجود دارد. همچنین زوج‌های (“I”,”d”) و (“w”,”id”) و (“wid”,”est_”) نیز با فرکانس ۳ قابل ترکیب هستند تا در نهایت به رشته ”widest_“ برسیم. دادگان جدید به شکل زیر می‌باشند:

Low_
2 Low e r _ V = {d,e,i,l,n,o,r,s,t,w,_,t_,st_,est_,lo,low,ew,new,low_,
3 Widest_ newest_,id,wid,widest_}
5 Newest_

در مراحل نهایی نیز زوج‌های (”r“,”_”) و (“e”,”r_”) و (”low“,”er_”) با فرکانس ۲ ترکیب می‌شوند. این حروف می‌توانند با هر ترتیبی ترکیب شوند تا به رشته ”lower_“ برسند. شکل نهایی لغت‌نامه و دادگان آموزشی در ادامه مشخص است:

5 Low_
2 Lower_ V = {d,e,i,l,n,o,r,s,t,w,_,t_,st_,est_,lo,low,ew,new,low_,
3 Widest_ newest_,id,wid,widest_,r_,er_,lower_}
5 Newest_

توکن‌سازی کلمه lowest:

با آموزش مدل خود به کمک داده‌های فوق و با توجه به اینکه کلمه ”lowest_“ در این لغت‌نامه وجود ندارد، به توکن‌های تقسیم می‌شود که در این مجموعه حضور دارند. در نتیجه کلمه ”lowest_“ به توکن‌های ”low“ و ”est_“ تقسیم می‌شود.

گام دوم:

در گام دوم مدل‌های WordPiece و BPE با کمک کتاب‌خانه Huggingface یک‌بار با استفاده از مجموعه داده کتاب Gutenberg و یک‌بار با استفاده از داده‌های ویکیپدیا آموزش می‌بینند. تعداد توکن‌های ایجاد شده توسط هر کدام از این ۴ توکن‌ساز در جدول زیر قابل مشاهده است:

Wiki	*Gutenberg*	
۷۷۷۳۶۶	۱۶۵۳۷	*BPE*
۸۱۲۹۸۶	۱۷۵۶۹	*WordPice*

روش پیاده سازی:

برای هر چهار توکن‌ساز به طور مشابهی ابتدا مدل تعریف می‌شود. سپس یک ترینر از نوع الگوریتم مورد نظر تعریف می‌شود و برای آن، توکن‌های خاص(مانند "[UNK]")، حداقل فرکانس زوج حروف و حداکثر اندازه لغت‌نامه مشخص می‌شوند. در مرحله پیش توکن‌سازی، متن به از روی فواصل به کلمات شکسته می‌شود تا از تشکیل توکن‌های بیش از یک کلمه جلوگیری شود. در آخرین مرحله نیز مدل با استفاده از ترینر تعریف شده و متن ورودی پیش‌توکن سازی شده آموزش می‌بینند.

توکن کردن متن نمونه:

تعداد توکن تولید شده برای متن نمونه توسط هر یک از ۴ مدل در حدول زیر مشخص است:

Wiki	*Gutenberg*	
۴۱	۵۵	*BPE*
۳۹	۵۲	*WordPiece*

متن نمونه به شکل زیر می‌باشد:

```
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?! 😊
```

خروجی مدل BPE با دادگان آموزشی کتاب گوتنبرگ برای متن نمونه بالا به شکل زیر می‌باشد:

```
['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]']
```

مدل WordPiece با داده‌های آموزشی کتاب گوتنبرگ برای متن نمونه فوق خروجی زیر را تولید می‌کند:

```
['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]']
```

خروجی مدل BPE با دادگان آموزشی ویکیپدیا برای متن نمونه بالا به شکل زیر می‌باشد:

```
['This', 'is', 'a', 'deep', 'learning', 'token', 'ization', 'tutorial', '.', 'Token', 'ization', 'is', 'the', 'first', 'step', 'in', 'a', 'deep', 'learning', 'NL', 'P', 'pipeline', '.', 'We', 'will', 'be', 'comparing', 'the', 'tokens', 'generated', 'by', 'each', 'token', 'ization', 'model', '.', 'Excited', 'much', '?', '!', '[UNK]']
```

مدل WordPiece با داده‌های آموزشی ویکیپدیا برای متن نمونه فوق خروجی زیر را تولید می‌کند:

```
['This', 'is', 'a', 'deep', 'learning', 'token', '##ization', 'tutorial', '.', 'Token', '##ization', 'is', 'the', 'first', 'step', 'in', 'a', 'deep', 'learning', 'NL', '##P', 'pipeline', '.', 'We', 'will', 'be', 'comparing', 'the', 'tokens', 'generated', 'by', 'each', 'token', '##ization', 'model', '.', 'Excited', 'much', '[UNK]']
```

کیفیت توکن‌ها و تاثیر حجم دادگان آموزشی:

می‌دانیم حجم داده‌های آموزشی و تنوع کلمات و توکن‌های موجود در آن‌ها تاثیر به سزایی در کیفیت توکن‌های تولید شده توسط توکنایزرها دارد. توکنایزرهایی که با کتاب گوتنبرگ با هر کدام از الگوریتم‌های BPE و WordPiece آموزش داده شده‌اند به دلیل کوچک بودن این مجموعه داده و کافی نبودن تعداد کلمات آن، دارای تعداد توکن‌های بیشتری نسبت به توکنایزرهای که با مجموعه داده‌های ویکیپدیا آموزش داده شده‌اند می‌باشد. این تفاوت تنها به افزایش تعداد توکن‌ها نمی‌انجامد بلکه باعث عدم تشخیص برخی کلمات که در دادگان آموزشی گوتنبرگ حضور ندارند(OOV) می‌شود و این دسته از کلمات را به توکن‌های کوچک‌تر و بعضا بی‌معنی تقسیم می‌کند، در نتیجه باعث کاهش کیفیت خروجی مدل‌های آموزش داده شده با مجموعه داده کوچکتر می‌شود.

مدل‌های آموزش داده شده با داده‌های آموزشی یکسان و مدل‌های توکنایزر متفاوت نیز تعداد توکن‌های متفاوتی برای متن نمونه ایجاد می‌کنند که البته این تعداد تفاوت زیادی ندارد و بخشی از اختلاف آنها به دلیل حذف علائم نگارشی مانند "!" و "؟" می‌باشد. دیگر تفاوت تعداد توکن‌ها برای متن نمونه در توکنایزرهای آموزش دیده با کتاب گوتنبرگ اتفاق می‌افتد. در این توکنایزرها کلمه ”Tokenization“ در الگوریتم BPE به ۴ توکن و در الگوریتم WordPiece به ۳ توکن تقسیم می‌شود که نشان می‌دهد رویکرد استفاده از Likelihood در الگوریتم WordPiece پاسخگویی بهتری نسبت به استفاده از فرکانس در BPE دارد و توکن‌های ایجاد شده‌ی بهتری دارد..(هر چند که هر دو توکنایزر آموزش دیده با کتاب گوتنبرگ به دلیل استفاده از مجموعه داده آموزشی ضعیف توکن‌های خوبی ایجاد نمی‌کند!)

توکن‌های OOV:

همانطور که گفتیم در هنگام توکن‌سازی برای کلماتی که در لغت‌نامه وجود نداشته باشند ممکن است ۲ اتفاق بیافتد. اگر امکان داشته باشد به توکن‌هایی کوچکتر از واحد کلمه تقسیم می‌شود در غیر این صورت با توکن ”[UNK]“ جایگزین می‌شوند. کلماتی ”tokenization“ ، ”pipeline“ ، ”nlp“ و ”excited“ در مدل‌های آموزش داده شده با کتاب گوتنبرگ و کلمات ”tokenization“ و ”nlp“ در مدل‌های آموزش داده شده با داده‌های ویکیپدیا جزو کلمات OOV هستند که به توکن‌های کوچکتر تقسیم شده‌اند. همچنین اموجی ”😊“ نیز جزو کلمات OOV است که با توکن ”[UNK]“ جایگزین می‌شود.

گام سوم:

تعداد توکن‌های خروجی هر کدام از ۴ مدل آموزش داده شده برای کتاب گوتنبرگ در جدول زیر قابل مشاهده است:

ردیف	نام الگوریتم استفاده شده برای توکنایزر	تعداد توکن‌های 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	BPE	۱۲۲۸۸۵	۱۲۷۳۷۷
۲	WordPiece	۱۲۲۸۰۵	۱۲۴۱۱۴