



Assignment NO.5 Solutions

NLP | Fall 1401 | Dr.Minayi

Student name : **Amin Fathi**

Student id : **400722102**

Problem 1

Sequence Models, مدل‌های یادگیری ماشینی هستند که توالی داده‌ها را وارد یا خروجی می‌کنند. داده‌های متوالی شامل جریان های متنی، کلیپ های صوتی، کلیپ های ویدئویی، داده های سری زمانی و غیره می باشد.

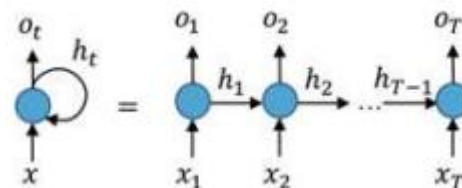
کاربردهای Sequence Models:

۱. تشخیص گفتار: در تشخیص گفتار، یک کلیپ صوتی به عنوان ورودی داده می‌شود و سپس مدل باید رونوشت متن خود را تولید کند. در اینجا هر دو ورودی و خروجی دنباله‌ای از داده‌ها هستند.

۲. طبقه‌بندی احساسات: در طبقه‌بندی احساسات، نظرات بیان شده در یک متن دسته‌بندی می‌شود. در اینجا ورودی دنباله‌ای از کلمات است.

۳. شناسایی فعالیت ویدیویی: در تشخیص فعالیت ویدیویی، مدل نیاز به شناسایی فعالیت در یک کلیپ ویدیویی دارد. کلیپ ویدیویی دنباله‌ای از فریم‌های ویدیویی است، بنابراین در صورت فعالیت ویدیویی، ورودی یک دنباله از داده‌ها است.

شبکه عصبی بازگشتی (RNN) یک الگوریتم یادگیری عمیق است و نوعی معماری شبکه عصبی مصنوعی است که برای پردازش داده‌های متوالی تخصصی است. RNN ها بیشتر در زمینه پردازش زبان طبیعی استفاده می‌شوند. RNN حافظه داخلی را حفظ می‌کند، به همین دلیل برای مشکلات یادگیری ماشینی که شامل داده‌های متوالی است بسیار کارآمد هستند. RNN ها همچنین در پیش بینی سری‌های زمانی نیز استفاده می‌شوند.



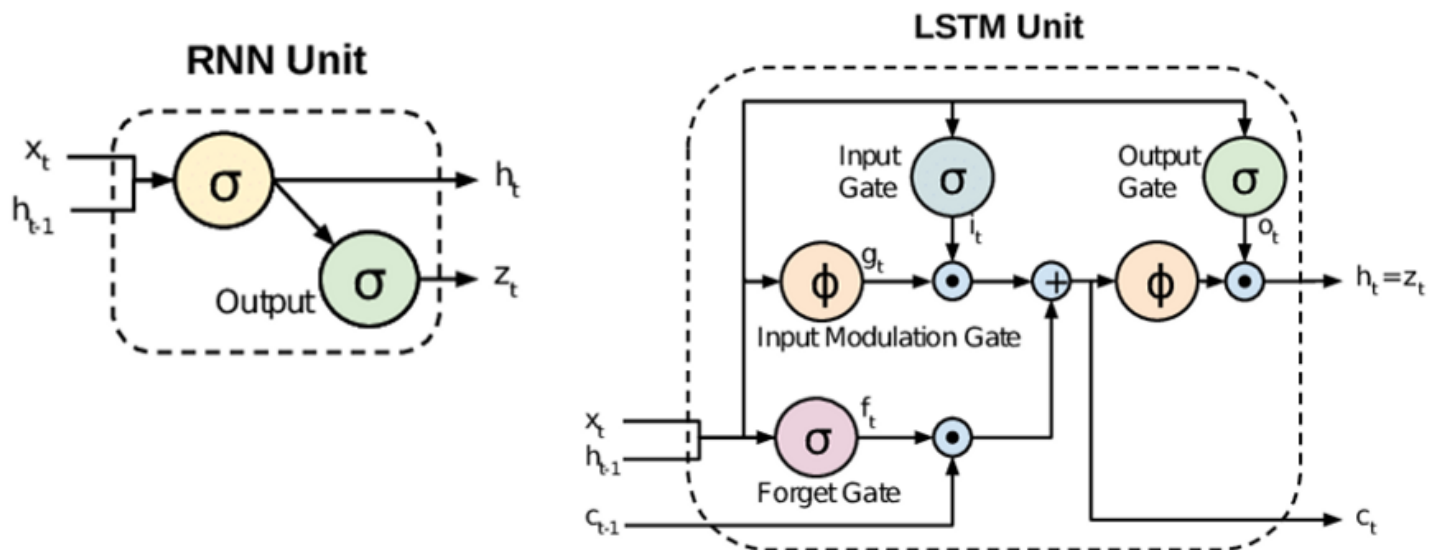
مزیت اصلی استفاده از RNN به جای شبکه‌های عصبی استاندارد این است که ویژگی‌ها در شبکه‌های عصبی استاندارد به اشتراک گذاشته نمی‌شوند. وزن‌ها در طول زمان در RNN به اشتراک گذاشته می‌شوند. RNN ها می‌توانند ورودی‌های قبلی خود را به خاطر بسپارند، اما شبکه‌های عصبی استاندارد قادر به یادآوری ورودی‌های قبلی نیستند. RNN اطلاعات تاریخی را برای محاسبه می‌گیرد.

حافظه بلند مدت کوتاه مدت (LSTM)

RNN های سنتی در گرفتن وابستگی های دوربرد خوب نیستند. این عمدتاً به دلیل مشکل گرادیان ناپدید شدن است. هنگام آموزش شیب های بسیار عمیق شبکه یا مشتقات به صورت تصاعدی کاهش می یابد زیرا در لایه ها پخش می شود. این به عنوان مشکل گرادیان ناپدید شدن شناخته می شود. از این گرادیان ها برای به روز رسانی وزن شبکه های عصبی استفاده می شود. هنگامی که گرادیان ها ناپدید می شوند، وزن ها به روز نمی شوند. گاهی اوقات می شود

به طور کامل آموزش شبکه عصبی را متوقف کنید. این مشکل گرادیان ناپدید شدن یک مسئله رایج در شبکه های عصبی بسیار عمیق است.

برای غلبه بر این مشکل ناپدید شدن گرادیان در RNN ها، حافظه کوتاه مدت بلند مدت توسط سب هوکرایتر و یورگن اشمیدهابر معرفی شد. LSTM تغییر در لایه پنهان RNN است. LSTM RNN ها را قادر می سازد تا ورودی های خود را در مدت زمان طولانی به خاطر بسپارند. در LSTM علاوه بر حالت پنهان، یک حالت سلولی به مرحله زمانی بعدی منتقل می شود.



LSTM می تواند وابستگی های دوربرد را ثبت کند. می تواند حافظه ورودی های قبلی را برای مدت زمان طولانی داشته باشد. در یک سلول LSTM 3 دروازه وجود دارد. دستکاری حافظه در LSTM با استفاده از این گیت ها انجام می شود. حافظه کوتاه مدت بلند مدت (LSTM) از گیت ها برای کنترل انتشار گرادیان در حافظه شبکه تکراری استفاده می کند.

منبع:

<https://towardsdatascience.com/sequence-models-and-recurrent-neural-networks-rnns-62cdeb4f1e1>

Problem 2

<S> Will mark hunt. <E> $\rightarrow n / v / v$

<S> Mark will hunt. <E> $\rightarrow n / m / v$

<S> Can Will mark? <E> $\rightarrow m / n / v$

<S> Mark can hunt. <E> $\rightarrow n / m / v$

a)

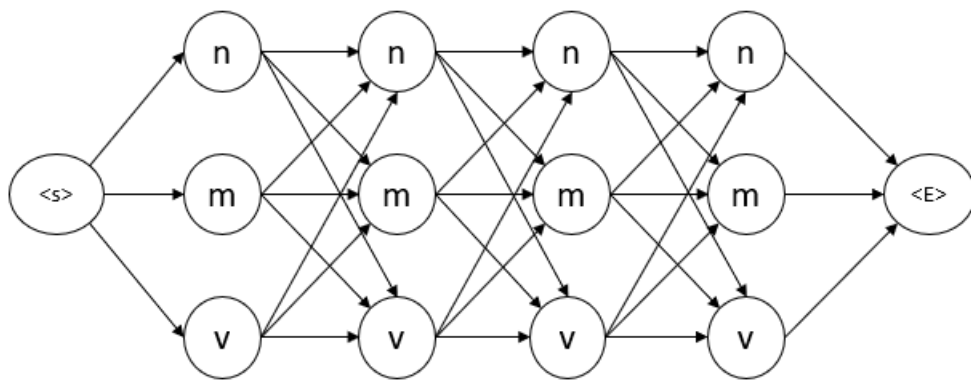
	Noun	Verb	Modat	Total count
Will	2	0	1	3
mark	2	2	0	4
hunt	1	2	0	3
can	0	0	2	2

b)

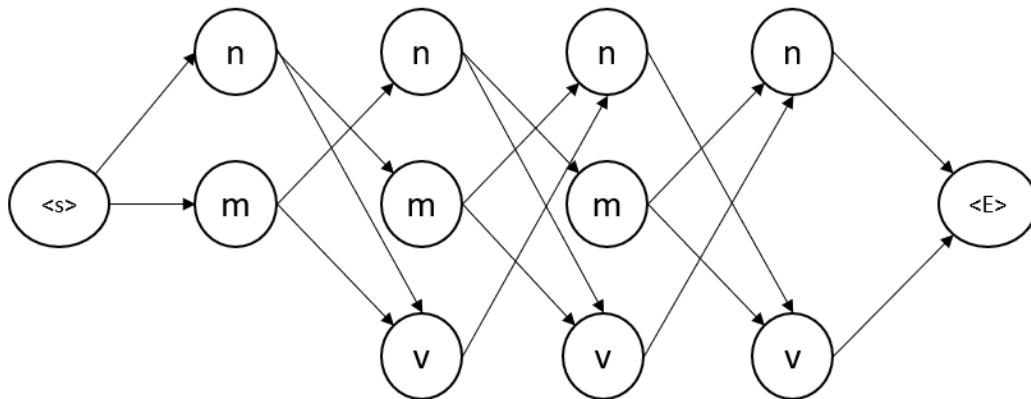
	<S>	Noun	Verb	Modal	<E>
<S>	0	$\frac{3}{4}$	0	$\frac{1}{4}$	0
Noun	0	0	$\frac{2}{5}$	$\frac{2}{5}$	$\frac{1}{5}$
Verb	0	$\frac{1}{4}$	0	0	$\frac{3}{4}$
Modal	0	$\frac{1}{3}$	$\frac{2}{3}$	0	0
<E>	0	0	0	0	0

c)

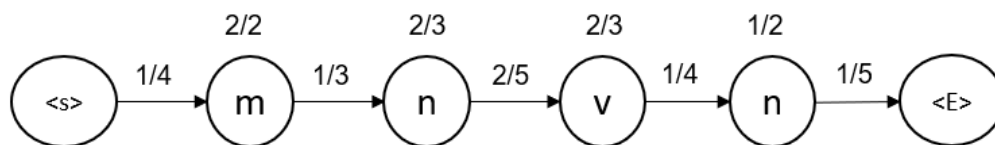
<S> Can will hunt Mark <E> $\rightarrow m / n / v / n$



یال های با وزن صفر را حذف می کنیم:



گره های با وزن صفر را حذف می کنیم:



احتمال نهایی به صورت زیر محاسبه می شود:


$$P = (1 \times 2 \times 1 \times 2 \times 2 \times 1 \times 1 \times 1) / (4 \times 2 \times 3 \times 3 \times 5 \times 3 \times 4 \times 2 \times 5) = 0.0003$$


Problem 3


پس از فراخوانی کتابخانه های لازم و بارگذاری دیتاست، جدول زیر که دیتاست بارگذاری شده است را نمایش می دهیم. دیتاست شامل برچسب های صفر تا ۵ است که میزان شباهت جفت جملات را نشان می دهد. صفر نشان دهنده ی کمترین شباهت و ۵ نشان دهنده ی بیشترین شباهت است.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datasets import load_dataset
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
import spacy
import textdistance
from sentence_transformers import SentenceTransformer
```


```
train_dataset = load_dataset('stsb_multi_mt', name='en', split='train')
test_dataset = load_dataset('stsb_multi_mt', name='en', split='test')
```


Downloading builder script: 100%  7.43k/7.43k [00:00<00:00, 326kB/s]


Downloading metadata: 100%  19.0k/19.0k [00:00<00:00, 22.5kB/s]


Downloading readme: 100%  9.98k/9.98k [00:00<00:00, 269kB/s]

Downloading and preparing dataset stsb_multi_mt/en to /root/.cache/huggingface/datasets/stsb_multi_mt/en/1.0.0/a5d260e4b7aa82d1a1

Downloading data files: 100%  3/3 [00:02<00:00, 1.05it/s]

Downloading data:  709k/? [00:00<00:00, 4.88MB/s]

Downloading data:  205k/? [00:00<00:00, 3.16MB/s]

Downloading data:  159k/? [00:00<00:00, 1.80MB/s]

Dataset stsb_multi_mt downloaded and prepared to /root/.cache/huggingface/datasets/stsb_multi_mt/en/1.0.0/a5d260e4b7aa82d1ab7379f

WARNING:datasets.builder:Found cached dataset stsb_multi_mt (/root/.cache/huggingface/datasets/stsb_multi_mt/en/1.0.0/a5d260e4b7aa82d1ab7379f)

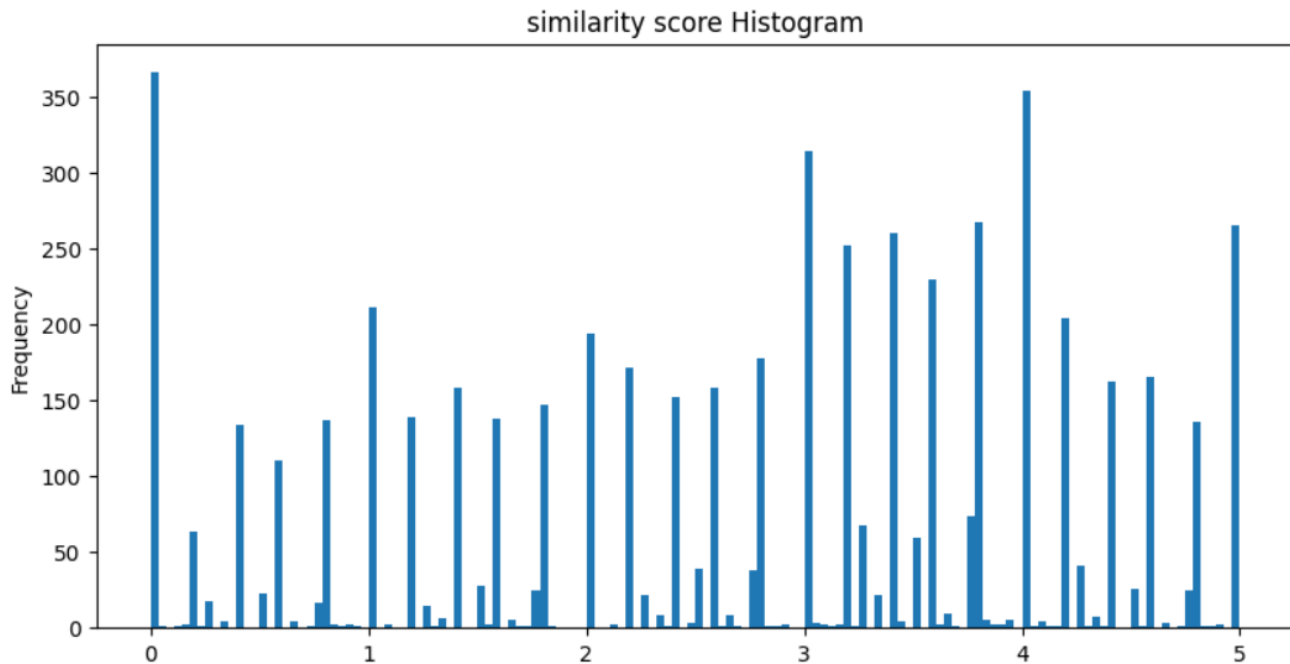
```
train_df = pd.DataFrame(train_dataset)
test_df = pd.DataFrame(test_dataset)
train_df
```

	sentence1	sentence2	similarity_score
0	A plane is taking off.	An air plane is taking off.	5.00
1	A man is playing a large flute.	A man is playing a flute.	3.80
2	A man is spreading shredded cheese on a pizza.	A man is spreading shredded cheese on an uncoo...	3.80
3	Three men are playing chess.	Two men are playing chess.	2.60
4	A man is playing the cello.	A man seated is playing the cello.	4.25
...
5744	Severe Gales As Storm Clodagh Hits Britain	Merkel pledges NATO solidarity with Latvia	0.00
5745	Dozens of Egyptians hostages taken by Libyan t...	Egyptian boat crash death toll rises as more b...	0.00
5746	President heading to Bahrain	President Xi: China to continue help to fight ...	0.00
5747	China, India vow to further bilateral ties	China Scrambles to Reassure Jittery Stock Traders	0.00
5748	Putin spokesman: Doping charges appear unfounded	The Latest on Severe Weather: 1 Dead in Texas ...	0.00

5749 rows × 3 columns

نمودار پراکندگی داده های train در شکل زیر نشان داده شده است.

```
plt.rcParams.update({'figure.figsize':(10,5), 'figure.dpi':100})
plt.hist(train_df.similarity_score, bins=140)
plt.gca().set(title='similarity score Histogram', ylabel='Frequency')
plt.show()
```



تابع زیر را برای پیش پردازش پیاده سازی می کنیم. این تابع متون را با کلمه نویسی، کوچک کردن، حذف اعداد و توقف کلمات پیش پردازش می کند. همچنین در ادامه تابع Jaccard پیاده سازی می شود. شمردن کلمات منحصر به فرد مشترک در دو متن، ساده ترین راه برای مقایسه است. اسناد طولانی تر، تعداد کلمات مشترک بیشتری دارند. برای غلبه بر این مشکل، از شباهت Jaccard استفاده می شود که فرمول آن به صورت زیر است.

$$Jaccard\ Similarity = \frac{Number\ of\ common\ unique\ words}{Total\ number\ of\ unique\ words}$$

پس از اعمال پیش پردازش و تابع Jaccard، خروجی به صورت زیر است. این خروجی ۵ نمونه با بیشترین و ۵ نمونه با کمترین شباهت را نشان می دهد.

```

nlp = spacy.load('en_core_web_sm')
def text_processing(sentence):
    """
    Lemmatize, lowercase, remove numbers and stop words

    Args:
        sentence: The sentence we want to process.

    Returns:
        A list of processed words
    """
    sentence = [token.lemma_.lower()
                 for token in nlp(sentence)
                 if token.is_alpha and not token.is_stop]

    return sentence

```

```

def jaccard_sim(row):
    s1 = row['clean_sentence1']
    s2 = row['clean_sentence2']

    return textdistance.jaccard.normalized_similarity(s1, s2)

```

```

train_df['jaccard_sim'] = train_df.apply(jaccard_sim, axis=1)
test_df['jaccard_sim'] = test_df.apply(jaccard_sim, axis=1)

```

```
test_df.sort_values(by=['jaccard_sim'], ascending=False)[['sentence1', 'sentence2', 'similarity_score', 'jaccard_sim']].head(5)
```

	sentence1	sentence2	similarity_score	jaccard_sim
512	Three women cook.	Two women cooking.	3.2	1.0
759	How do you do that?	How should you do that?	4.0	1.0
60	The man is kissing and hugging the woman.	A man is hugging and kissing a woman.	5.0	1.0
727	You answered your own question.	You've answered your own question already.	5.0	1.0
737	Can you do this?	Can you do it?	5.0	1.0

```
test_df.sort_values(by=['jaccard_sim'], ascending=False)[['sentence1', 'sentence2', 'similarity_score', 'jaccard_sim']].tail(5)
```

	sentence1	sentence2	similarity_score	jaccard_sim
678	You need to read a lot to know what you like a...	Yes, you should create a portfolio site to sho...	0.0	0.0
675	There is no maximum.	There is no quarantine period.	0.0	0.0
72	A woman opens a window.	A man is crawling.	0.0	0.0
874	There are two possible causes for this:	There are two options for you -	1.0	0.0
126	Someone is drawing.	Someone is dancing.	0.3	0.0

در ادامه تابع `cos_sim` پیاده سازی شده است. پس از محاسبه ی `embedding`، نهایتاً خروجی برای ۵ نمونه با بیشترین شباهت و ۵ نمونه با کمترین شباهت حاصل می شود.

```
def cos_sim(sentence1_emb, sentence2_emb):
    """
    Cosine similarity between two columns of sentence embeddings
    """
    cos_sim = cosine_similarity(sentence1_emb, sentence2_emb)
    return np.diag(cos_sim)
```

```
tfidf_model = TfidfVectorizer(lowercase=True, stop_words='english')
X_train = pd.concat([train_df.sentence1, train_df.sentence2]).unique()

tfidf_model.fit(X_train)
```

```
train_emb1 = tfidf_model.transform(train_df.sentence1)
train_emb2 = tfidf_model.transform(train_df.sentence2)
test_emb1 = tfidf_model.transform(test_df.sentence1)
test_emb2 = tfidf_model.transform(test_df.sentence2)
```

```
train_df['tfidf_sim'] = cos_sim(train_emb1, train_emb2)
test_df['tfidf_sim'] = cos_sim(test_emb1, test_emb2)
```

```
test_df.sort_values(by=['tfidf_sim'], ascending=False)[['sentence1', 'sentence2', 'similarity_score', 'tfidf_sim']].head(5)
```

	sentence1	sentence2	similarity_score	tfidf_sim
1324	On War Criminals and Heroes: The Whitewashing ...	On war criminals and heroes: The whitewashing ...	5.000	1.0
648	The rule - When in doubt throw it out!	I always go by the rule "When in doubt, throw ...	4.000	1.0
1070	"I expect Japan to keep conducting interventio...	Junya Tanase, forex strategist at JP Morgan Ch...	4.733	1.0
33	A cat is eating some corn.	A cat is eating corn on the cob.	4.250	1.0
60	The man is kissing and hugging the woman.	A man is hugging and kissing a woman.	5.000	1.0

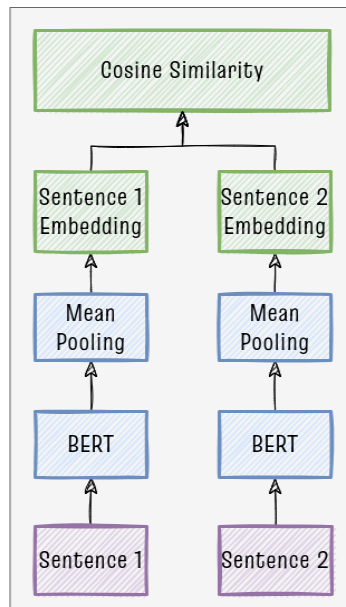
```
test_df.sort_values(by=['tfidf_sim'], ascending=False)[['sentence1', 'sentence2', 'similarity_score', 'tfidf_sim']].tail(5)
```

	sentence1	sentence2	similarity_score	tfidf_sim
546	A woman is posing for a Christmas card.	A girl is taking a photo.	1.20	0.0
97	Someone typed on a keyboard.	Someone is typing.	4.50	0.0
95	The lady peeled the potatoe.	A woman is peeling a potato.	4.75	0.0
323	Two black and white dogs at the bottom of stairs.	A dog is lying at the bottom of a staircase ne...	2.60	0.0
242	A man is doing exercise.	A baby is laughing.	0.00	0.0

در این قسمت از معماری شکل زیر استفاده شده است. ایده اصلی این شبکه به این صورت است که از مجموعه داده برچسب‌دار به‌عنوان داده‌های آموزشی استفاده می‌کند. برای هر متن در مجموعه آموزشی، جاسازی‌های متنی کلمات آن متن را با استفاده از مدل BERT از پیش آموزش دیده به عنوان رمزگذار محاسبه می‌کند. میانگین عنصری همه جاسازی‌های نشانه را محاسبه کرد تا یک جمله با بعد ثابت برای کل متن به دست آورد. این عملیات Mean Pooling نام دارد.

هدف این مدل این است که جاسازی‌ها را برای متون مشابه به یکدیگر نزدیک‌تر کند؛ به‌طوری که فاصله بین آن‌ها نزدیک به صفر باشد. برعکس، مدل قصد دارد جاسازی‌ها از متون غیرمشابه از یکدیگر دوتر کند به‌طوری که فاصله بین آن‌ها زیاد باشد. پس از تکمیل آموزش مدل، می‌توان هر دو متن را با محاسبه‌ی شباهت بین جاسازی‌های آن دو متن مقایسه کرد. مدل Bi-Encoder یک

متن را در یک زمان به عنوان ورودی می‌گیرد و یک بردار تعبیه شده با ابعاد ثابت را به عنوان خروجی نشان می‌دهد. سپس می‌توان هر دو سند را با محاسبه شباهت بین جاسازی‌های آن دو مقایسه کرد.



این معماری به صورت زیر پیاده سازی می‌شود.

```

transformer_model = SentenceTransformer('stsb-mpnet-base-v2')

train_emb1 = transformer_model.encode(train_df.sentence1, show_progress_bar=True)
train_emb2 = transformer_model.encode(train_df.sentence2, show_progress_bar=True)
test_emb1 = transformer_model.encode(test_df.sentence1, show_progress_bar=True)
test_emb2 = transformer_model.encode(test_df.sentence2, show_progress_bar=True)

train_df['nn_sim'] = cos_sim(train_emb1, train_emb2)
test_df['nn_sim'] = cos_sim(test_emb1, test_emb2)
  
```

```
test_df.sort_values(by=['nn_sim'], ascending=False)[['sentence1', 'sentence2', 'similarity_score', 'nn_sim']].head(5)
```

	sentence1	sentence2	similarity_score	nn_sim
805	What are your goals?	What are you goals?	5.0	0.992782
623	A brown dog is jumping.	A brown dog is jumping	5.0	0.992404
566	People gathered in a room.	People gathered together in a room.	5.0	0.991575
427	a dog jumps into the water.	A dog is jumping into the water.	5.0	0.990334
136	A woman plays the flute.	A woman is playing the flute.	5.0	0.990013

```
test_df.sort_values(by=['nn_sim'], ascending=False)[['sentence1', 'sentence2', 'similarity_score', 'nn_sim']].tail(5)
```

	sentence1	sentence2	similarity_score	nn_sim
242	A man is doing exercise.	A baby is laughing.	0.0	-0.113115
156	A person is playing a piano.	A person is slicing a potato.	0.5	-0.113264
277	Two men standing in grass staring at a car.	A woman in a pink top posing with beer.	0.2	-0.132677
68	A man is playing a guitar.	A woman is riding a horse.	0.5	-0.148015
445	A brown and white dog is running across a brow...	A person hanging from a rocky cliff.	0.0	-0.211263

علیرغم عملکرد قوی در مجموعه داده STSB، متأسفانه Transformers جملات، مدل‌هایی کاملاً تحت نظارت هستند و برای آموزش به مجموعه بزرگی از جفت جملات نیاز دارند. بنابراین، استفاده از Transformers جمله در حوزه‌های جدید فرآیندی زمان‌بر و پرهزینه برای جمع‌آوری داده‌های برچسب‌گذاری‌شده عظیم و با کیفیت بالا است.

در ادامه مقایسه‌ای صورت گرفته و طبق نتایج مشاهده شده، شبکه عصبی عملکرد بهتری را دارا است. در داده‌های test، jaccard و در داده‌های train، tfidf نتایج خوبی را نشان داده‌اند. دلیل برتری شبکه عصبی، نبود شرط مشابه بودن جملات مشابه با تعداد کمات مشترک زیاد است. شرطی که در دو حالت دیگر محدودکننده بوده و باعث کاهش عملکرد آن شده است.

```
score_cols = ['similarity_score', 'jaccard_sim', 'tfidf_sim', 'nn_sim']

train_spearman_rank_corr = train_df[score_cols].corr(method='spearman').iloc[1:,0:1]*100
test_spearman_rank_corr = test_df[score_cols].corr(method='spearman').iloc[1:,0:1]*100
```

```
print(train_spearman_rank_corr)
print(test_spearman_rank_corr)
```

```
          similarity_score
jaccard_sim      66.409271
tfidf_sim       68.191136
nn_sim          96.558008
          similarity_score
jaccard_sim      66.026529
tfidf_sim       61.420989
nn_sim          88.572413
```

منبع:

<https://towardsdatascience.com/semantic-textual-similarity-83b3ca4a840e>