Universal Sentence Encoder

2019. 10. 19
발표: 염혜원

https://arxiv.org/abs/1803.11175

1 Introduction

- NLP에서, 워드 레벨의 pre-trained Embedding 을 활용한 성능 향상이 있어왔음.
- 최근의 연구는 문장 레벨의 pre-trained Embedding이 strong performance를 나타냄 (Conneau et al., 2017).
- 본 연구에서는 두 가지 모델의 Sentence Embedding를 제시하였고 이를 Transfer Learning에 활용해 적은 수의 트레이닝 데이터셋으로 매우 높은 성능을 달성함

2 Model Toolkit

- Sentence Embedding 벡터 생성을 위한 두 가지 모델
  - Transformer-based
  - DAN(Deep Averaging Network)
- TF Hub에서 다운받을 수 있음
  - https://tfhub.dev/google/universal-sentence-encoder/2

```
[ ]  module_url = "https://tfhub.dev/google/universal-sentence-encoder
```

module_url: https://tfhub.dev/google/universal-sentence-encoder/2

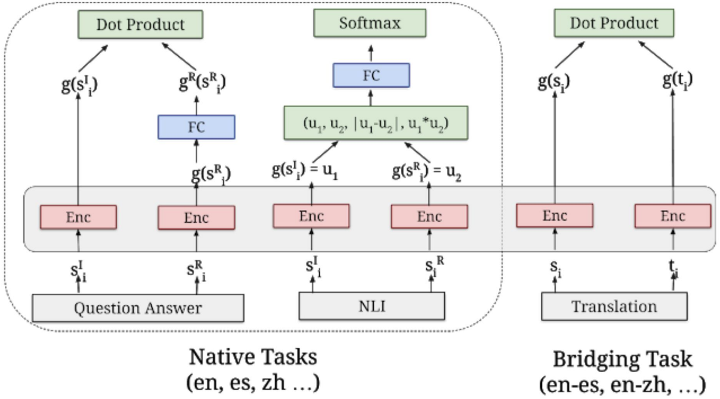
```
[ ]
# Import the Universal Sentence Encoder's TF Hub module
embed = hub.Module(module_url)

# Compute a representation for each message, showing various lengths supported.
word = "Elephant"
sentence = "I am a sentence for which I would like to get its embedding."
paragraph = (
    "Universal Sentence Encoder embeddings also support short paragraphs. "
    "There is no hard limit on how long the paragraph is. Roughly, the longer "
    "the more 'diluted' the embedding will be.")
messages = [word, sentence, paragraph]

# Reduce logging output.
tf.logging.set_verbosity(tf.logging.ERROR)

with tf.Session() as session:
  session.run([tf.global_variables_initializer(), tf.tables_initializer()])
  message_embeddings = session.run(embed(messages))

  for i, message_embedding in enumerate(np.array(message_embeddings).tolist()):
    print("Message: {}".format(messages[i]))
    print("Embedding size: {}".format(len(message_embedding)))
    message_embedding_snippet = ", ".join(
        (str(x) for x in message_embedding[:3]))
    print("Embedding: [{}, ...]\n".format(message_embedding_snippet))
```

3 Encoders

- Transformer의 Encoder sub-graph
  - 해당 sub-graph는 단어들 간의 순서와 문맥 정보를 포함시키기 위해 attention을 활용함
  - 문맥 정보가 포함된 단어의 representation들을 element-wise sum하여 fixed-length sentence encoding vector 얻음 (이때 문장 길이의 영향을 상쇄하기 위해 sqrt[문장같이]로 나누어 줌)
- DAN(Deep Averaging Network)
  - Averaging words/bi-grams embeddings 이후 Feedforward DNN에 태워 문장 임베딩 벡터 생성
  - DNN은 계산 복잡도 측면의 이점 보유: 문장 길이에 대해 linear하게 증가

3 Encoders

- 인코더는 PTB(Penn Tree Bank) tokenized string을 인풋으로 받고 512-d 벡터 (Sentence Embedding) 반환
- Multi-task Learning
  - Generally 활용할 수 있는 문장 임베딩을 얻기 위해 여러 다운스트림 task에 적용
    : Skip-Thought like task, Conversational input-response task, Classifications



Multi-task training structure of the Universal Sentence Encoder. A variety of tasks and task structures are joined by shared encoder layers/parameters (pink boxes).

4 Transfer Tasks

Dataset	Train	Dev	Test	
SST	67,349	872	1,821	Sentiment Classification
STS Bench	5,749	1,500	1,379	Semantic textual similarity
TREC	5,452	-	500	Question Classification
MR	-	-	10,662	Movie Review Classification(5-scale)
CR	-	-	3,775	Customer Review Classification
SUBJ	-	-	10,000	Subjectivity of Sentences
MPQA	-	-	10,606	Phrase level opinion polarity

Table 1: Transfer task evaluation sets

4 Transfer Tasks

- **WEAT**
  - Implicit Association Test(IAT)에 대해 심리학 문헌으로부터 추출한 단어쌍 (Word pairs from the psychology literature on implicit association tests)
  - **Model bias를 정의하기 위한 용도로 활용**

5 Transfer Learning Models

- 분류 Task: Transformer/DAN 인코더의 output을 task specific DNN에 제공
- Semantic Similarity Task: 아래 식을 이용해 sentence embedding들 간의 유사도를 직접 계산

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \left(1 - \arccos\left(\frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \; ||\mathbf{v}||}\right)/\pi\right) \quad (1)$$

[5] We find that using a similarity based on angular distance performs better on average than raw cosine similarity.

5 Transfer Learning Models

- Baselines: 각 Task에 대해 word level transfer / no transfer learning 의 두 모델을 베이스라인으로 삼음
- Combined Transfer Models: 문장 임베딩과 단어 임베딩을 함께 사용
    - 두 가지 representation을 결합(Concat) 후 transfer task classification layer로 전달

6 Experiments

1. Training Dataset Size에 따른 Transfer Learning 효과성 탐색
2. Transformer Encoder vs. DAN Encoder (모델 복잡도와 정확도 관점)
3. WEAT word list를 활용한 model bias 측정

7 Results

- Transformer > DAN
- Sentence & Word 임베딩 > Sentence 임베딩 > Word 임베딩

Model	MR	CR	SUBJ	MPQA	TREC	SST	STS Bench (dev / test)
Sentence & Word Embedding Transfer Learning							
USE_D+DAN (w2v w.e.)	77.11	81.71	93.12	87.01	94.72	82.14	–
USE_D+CNN (w2v w.e.)	78.20	82.04	93.24	85.87	97.67	85.29	–
USE_T+DAN (w2v w.e.)	81.32	86.66	93.90	88.14	95.51	86.62	–
USE_T+CNN (w2v w.e.)	81.18	87.45	93.58	87.32	98.07	86.69	–
Sentence Embedding Transfer Learning							
USE_D	74.45	80.97	92.65	85.38	91.19	77.62	0.763 / 0.719 (r)
USE_T	81.44	87.43	93.87	86.98	92.51	85.38	0.814 / 0.782 (r)
USE_D+DAN (lrn w.e.)	77.57	81.93	92.91	85.97	95.86	83.41	–
USE_D+CNN (lrn w.e.)	78.49	81.49	92.99	85.53	97.71	85.27	–
USE_T+DAN (lrn w.e.)	81.36	86.08	93.66	87.14	96.60	86.24	–
USE_T+CNN (lrn w.e.)	81.59	86.45	93.36	86.85	97.44	87.21	–
Word Embedding Transfer Learning							
DAN (w2v w.e.)	74.75	75.24	90.80	81.25	85.69	80.24	–
CNN (w2v w.e.)	75.10	80.18	90.84	81.38	97.32	83.74	–
Baselines with No Transfer Learning							
DAN (lrn w.e.)	75.97	76.91	89.49	80.93	93.88	81.52	–
CNN (lrn w.e.)	76.39	79.39	91.18	82.20	95.82	84.90	–

7 Results

- Data가 작을 수록, sentence level transfer learning이 성능이 word level 대비 우수
- Transformer 인코더의 경우 1k 데이터 만으로도 다른 모델 w/ 67.3k 데이터 수준

Model	SST 1k	SST 2k	SST 4k	SST 8k	SST 16k	SST 32k	SST 67.3k
Sentence & Word Embedding Transfer Learning							
USE_D+DNN (w2v w.e.)	78.65	78.68	79.07	81.69	81.14	81.47	82.14
USE_D+CNN (w2v w.e.)	77.79	79.19	79.75	82.32	82.70	83.56	85.29
USE_T+DNN (w2v w.e.)	85.24	84.75	85.05	86.48	86.44	86.38	86.62
USE_T+CNN (w2v w.e.)	84.44	84.16	84.77	85.70	85.22	86.38	86.69
Sentence Embedding Transfer Learning							
USE_D	77.47	76.38	77.39	79.02	78.38	77.79	77.62
USE_T	84.85	84.25	85.18	85.63	85.83	85.59	85.38
USE_D+DNN (lrn w.e.)	75.90	78.68	79.01	82.31	82.31	82.14	83.41
USE_D+CNN (lrn w.e.)	77.28	77.74	79.84	81.83	82.64	84.24	85.27
USE_T+DNN (lrn w.e.)	84.51	84.87	84.55	85.96	85.62	85.86	86.24
USE_T+CNN (lrn w.e.)	82.66	83.73	84.23	85.74	86.06	86.97	87.21
Word Embedding Transfer Learning							
DNN (w2v w.e.)	66.34	69.67	73.03	77.42	78.29	79.81	80.24
CNN (w2v w.e.)	68.10	71.80	74.91	78.86	80.83	81.98	83.74
Baselines with No Transfer Learning							
DNN (lrn w.e.)	66.87	71.23	73.70	77.85	78.07	80.15	81.52
CNN (lrn w.e.)	67.98	71.81	74.90	79.14	81.04	82.72	84.90

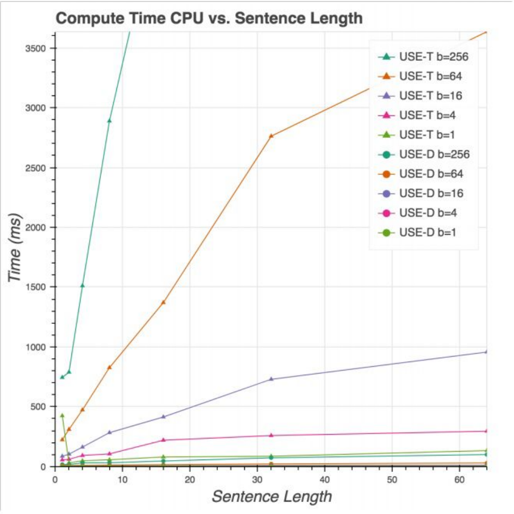
7 Results

- Caliskan et al. (2017)에서, GloVe(w/ DAN) 인코더로 단어간의 association을 재현한 것과 마찬가지로, 본 연구에서 제시하는 인코더도 pleasantness vs. unpleasantness와 같은 단어간의 assocation을 나타냄
- 다만, 본 모델의 경우 ageism, racism, sexism을 반영하는 단어쌍에 대해서는 약한 관계를 보임
  - 학습 데이터 구성, 다운스트림 task mixture의 차이가 그 원인이 될 수 있음

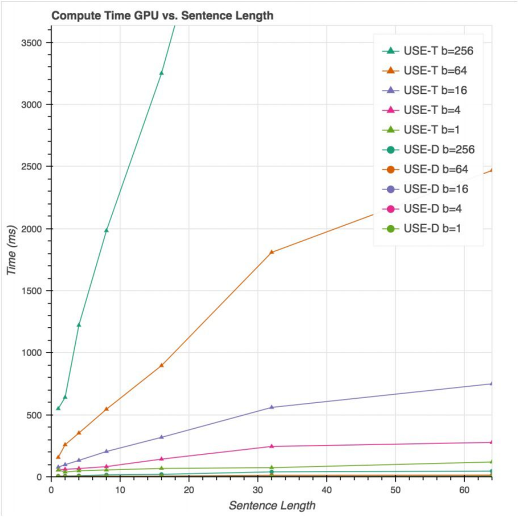
Target words	Attrib. words	Ref	GloVe		Uni. Enc. (DAN)	
			d	p	d	p
Eur.-American vs Afr.-American names	Pleasant vs. Unpleasant 1	a	1.41	10^{-8}	0.361	0.035
Eur.-American vs. Afr.-American names	Pleasant vs. Unpleasant from (a)	b	1.50	10^{-4}	-0.372	0.87
Eur.-American vs. Afr.-American names	Pleasant vs. Unpleasant from (c)	b	1.28	10^{-3}	0.721	0.015
Male vs. female names	Career vs family	c	1.81	10^{-3}	0.0248	0.48
Math vs. arts	Male vs. female terms	c	1.06	0.018	0.588	0.12
Science vs. arts	Male vs female terms	d	1.24	10^{-2}	0.236	0.32
Mental vs. physical disease	Temporary vs permanent	e	1.38	10^{-2}	1.60	0.0027
Young vs old peoples names	Pleasant vs unpleasant	c	1.21	10^{-2}	1.01	0.022
Flowers vs. insects	Pleasant vs. Unpleasant	a	1.50	10^{-7}	1.38	10^{-7}
Instruments vs. Weapons	Pleasant vs Unpleasant	a	1.53	10^{-7}	1.44	10^{-7}

8 Resource Usage

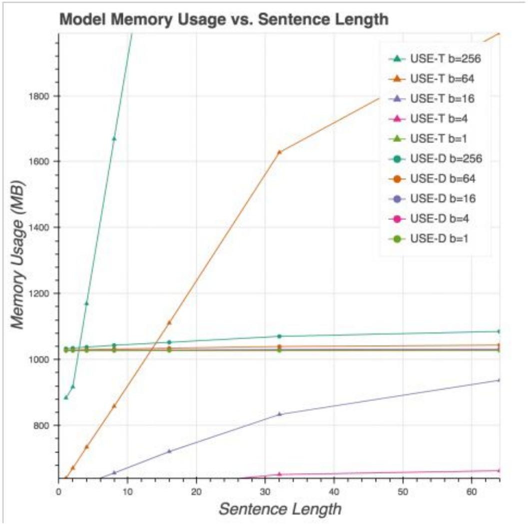
- Transformer 모델 계산복잡도: O(n^2)
- DAN 모델 계산복잡도: O(n)



(a) CPU Time vs. Sentence Length



(b) GPU Time vs. Sentence Length



(c) Memory vs. Sentence Length

9 Conclusion

- Sentence level 임베딩을 활용하는 경우 word level 임베딩 만을 사용한 경우 대비 transfer learning 성능이 높음 (Sentence + Word 의 경우 최고 성능)
- 두 인코딩 모델(Transformer vs. DAN)은 accuracy와 complexity측면의 trade-off 존재하며 인코딩 모델을 선택할 때 고려되어야 함