

시니어 개발자의 인공지능 도전기

헬로엔엠에스
모두의연구소 - 자연어 처리반
현청천

간단한 자기 소개

- 개발경력 10년 이상의 40대 시니어 개발자
 - 헬로엔엠에스
 - Front end, Back end, Mobile
- 나의 인공지능 경력 (5개월)
 - 2018년 8월부터 시작
 - 1개월간 책1권, YouTube 강의를 통해 독학
 - 모두의연구소 자연어 처리반에 문을 두드림
 - 너무많은 수학들, 용어들, 새로운 기술로 인한 어려움
- 참여동기
 - 자연어 처리반 내에서 도전팀 모집한다고 해서 얼떨결에
 - 그냥 대회라는 것이 어떤 곳인가 궁금해서

Base Model

- Embedding + Bi-Rnn + CRF가 결합된 개선 포인트가 잘 보이지 않는 모델
- 로컬 PC에서도 잘 돌아가는 모델
 - 알고보니 데이터 50개만 사용
- 전체 9만개 데이터로 내가 돌리면 85 베이스라인은 88 이유가 뭘까? 원인을 알 수 없음
- 일단 요즘 뜨는 Elmo, BERT 등을 적용해 보자

Embedding 1/2

- nsml 환경에서는 pre-train된 embedding 모델 적용 불가
- Tensorflow hub의 한국어 모델 4가지를 시험 함
<https://tfhub.dev/s?language=ko&module-type=text-embedding>
- Tensorflow 버전 업 설정. /tmp/THUB에 다운로드 함
- Base model 보다는 성능이 좋아 지지만 baseline을 넘지는 못함
87.1

Embedding 2/2

1,000	nnlm-ko-dim50-with-normalization[O, O]	-		19	0.6245945609	기본모델에 nnlm-ko-dim50-with-normalization 내부 테스트를 진행 하였습니다. (epoch을 좀더 늘려봐야 할 듯 합니다.)
1,000	nnlm-ko-dim50-with-normalization[O, O]	-		39	0.6055630979	기본모델에 nnlm-ko-dim50-with-normalization epoch 40 적용
10,000	nnlm-ko-dim50-with-normalization[O, O]	0.712587		9	0.8032543003	기본모델에 nnlm-ko-dim50-with-normalization data 10,000 epoch 10 적용
10,000	nnlm-ko-dim50-with-normalization[O, O]	0.724077		19	0.8045136187	기본모델에 nnlm-ko-dim50-with-normalization data 10,000 epoch 20 적용
10,000	nnlm-ko-dim50[O, O]	0.72144		19	0.8030570611	기본모델에 nnlm-ko-dim50 data 10,000 epoch 20 적용
10,000	nnlm-ko-dim128-with-normalization[O, O]	0.715079		19	0.7762478215	기본모델에 nnlm-ko-dim128-with-normalization data 10,000 epoch 20 / lstm_units=64 / char_lstm_units=64
10,000	nnlm-ko-dim128[O, O]	0.715404		19	0.7794605334	기본모델에 nnlm-ko-dim128 data 10,000 epoch 20 / lstm_units=64 / char_lstm_units=64
10,000	-	0.697071		19	0.7770715596	기본모델
10,000	nnlm-ko-dim50-with-normalization[O, O]	0.81813		19	0.8596301651	기본모델에 nnlm-ko-dim50-with-normalization data 80,000 epoch 20 적용
10,000	nnlm-ko-dim50-with-normalization[O, O]	0.822914		18	0.8669886881	기본모델에 nnlm-ko-dim50-with-normalization data 80,000 epoch 20 적용
10,000		0.776008		19	0.8172213814	기본모델 data 80,000 epoch 20 적용
9,000	nnlm-ko-dim50[O, O]	0.826657		18	0.8697291738	기본모델에 nnlm-ko-dim50 data 80,000 epoch 20 적용
9,000	nnlm-ko-dim128-with-normalization[O, O]	0.816373		9	0.8560903809	기본모델에 nnlm-ko-dim128-with-normalization data 80,000 epoch 20 적용
9,000	nnlm-ko-dim128[O, O]	0.821778		8	0.8671578686	기본모델에 nnlm-ko-dim128 data 80,000 epoch 20 적용
8,000	elmo[O, X]	0.821803		19		기본모델에 elmo data 82,000 epoch 20 적용
8,000	nnlm-ko-dim50-with-normalization[O, X]	0.81269		12	0.8606359271	기본모델에 nnlm-ko-dim50-with-normalization data 82,000 epoch 20 적용
8,000	nnlm-ko-dim50[O, X]	0.812046		6	0.8649234053	기본모델에 nnlm-ko-dim50 data 82,000 epoch 20 적용
8,000	nnlm-ko-dim128-with-normalization[O, X]	0.806027		9	0.8565788643	기본모델에 nnlm-ko-dim128-with-normalization data 82,000 epoch 20 적용
8,000	nnlm-ko-dim128[O, X]	0.802408		9	0.8575851877	기본모델에 nnlm-ko-dim128 data 82,000 epoch 20 적용
8,000	nnlm-ko-dim50[O, X]	0.829387		17	0.8659655891	기본모델에 nnlm-ko-dim140 data 82,000 epoch 20 char 50 batch 500
8,000	nnlm-ko-dim50[O, O]	0.823527		19	0.8635677023	기본모델에 nnlm-ko-dim140 data 82,000 epoch 20 char 50 batch 500
8,000	nnlm-ko-dim50-with-normalization[O, O]	0.827638		8	0.8700245117	기본모델에 nnlm-ko-dim140 data 82,000 epoch 20 char 50 batch 500
2,500	nnlm-ko-dim50[O, O]	0.88139		6	0.8692152292	기본모델에 nnlm-ko-dim140 data 82,000 epoch 20 char 50 batch 500 length 160
		0.880948		12	0.870074648	
		0.877485		25	0.8651240032	
2,500	nnlm-ko-dim50-with-normalization[O, O]	0.883312		5	0.8710598124	data 82,000 epoch 20 char 50 batch 500 length 160
2,500	nnlm-ko-dim50[O, O]	0.881315		8	0.8668291673	기본모델에 nnlm-ko-dim140 data 82,000 epoch 20 char 50 batch 500 length 160

CNN 1/2

- 대회 기간중 자연어 처리반 중현님의 ConvS2S(<https://arxiv.org/pdf/1705.03122.pdf>) 논문 발표를 듣고 CNN을 적용해 보기로 결정 함
- 이전에 적용한 embedding은 제외 한 후 base model에 CNN만을 적용 함
- Word, Char, Word+Char 3가지 경우로 테스트를 해 봄
- Word만 적용한 경우가 가장 좋았으나 baseline을 넘지는 못함
87.4
- 학습이 느리게 진행 됨

CNN 2/2

모델	Check Point	내부점수	등록점수	설명
bible/NER/79			0.8163099797	기본모델
bible/NER/80	9	0.78391	0.8443218852	CNN: [word]
	21	0.800858	0.8635929849	
	33	0.807464	0.8677061112	
	48	0.810524	0.8693284084	
	62	0.812345	0.8721456185	
	99	0.816009	0.8714529968	
bible/NER/83	99	0.817685	0.8743123149	CNN: [word], RELU: [word]
bible/NER/87	12	0.78557	0.8467658897	CNN: [word], RELU: [word], batch: 100
	26	0.796291	0.8600755	
bible/NER/108				(CNN, RELU) x 3
bible/NER/113	6	0.83861	0.8316868767	(CNN, normal, RELU) x 3 word & char

Self Attention

- Self Attention도 적용해보기로 함
- Training 결과는 좋으나 실제 적용하면 결과가 안좋음.
Overfit 되는 경향을 보임
- 시간관계상 많은 내용을 확인하지는 못함

Complex 2/2

- 마지막 5일은 이전 3가지 (Embedding, CNN, Attention)를 조합해서 가장 좋은 결과를 찾기로 결정
- 목표는 baseline 이라도 넘자
- Embedding+CNN+Attention, Embedding+CNN, Embedding+Attention, CNN+Attention 4가지 조합을 1,000 epoch 시도
- Embedding+CNN > Embedding+CNN+Attention 의 결과를 보였고 나머지 두개는 의미없는 결과
88.33

Complex 2/2

모델	Check Point	내부점수	등록점수	설명	
bible/NER/130				embedding + cnn(w)3 + self_attention, runrate 0.05, epoch 1000	stop at 441
	154	0.88232	0.8750994431		
	214	0.88514	0.8772764998		
	328	0.88323	0.8785724331		
bible/NER/131				embedding + cnn(w)3, runrate 0.05, epoch 1000	stop at 501
	76	0.88463	0.879547874		
	131	0.88705	0.8800785145		
	167	0.88734	0.8817264661		
	201	0.88778	0.8808332038		
	267	0.88805	0.881469375		
	327	0.88814	0.8808112858		
	366	0.88874	0.8818034119		
	370	0.88896	0.8833482024		
	501	0.88763	0.8817906272		
bible/NER/132				embedding + self_attention, runrate 0.05, epoch 1000	0.86273 이후 지속적으로 감소 함, 100 epoch 이 후 중지
	82	0.86273	0.8553890256		
bible/NER/133				cnn(w)3 + self_attention, runrate 0.05, epoch 1000	stop at 976
	228	0.88098	0.8731492094		
	239	0.88135	0.873736105		
	588	0.88251	0.8764571679		
bible/NER/140				embedding + cnn(w)3+normalize, runrate 0.05, epoch 1000	오히려 성능이 더 안나옴

Transformer 1/2

- 자연어 처리반 고수들이 작성한 자연어 책 (**텐서플로우와 자연어처리로 시작하는 자연어처리**) 리뷰 중 소개된 Transformer도 한번 적용 해 보기로 함
- Transformer + CRF를 사용
- 학습이 진행 될 수록 학습성능을 좋아지나 평가성능이 안좋아지는 현상 발생.
 - 심한 Overfit 발생
 - Shuffle을 추가 하기도 해서 일부 개선이 되었지만 비슷한 흐름을 보임

Transformer 2/2

설명				
encoder (word) / layer1	0.58782			
encoder (word, ne_dic) / layer1	0.80359	0.7997372989		
encoder (word, ne_dic, character) / layer1	0.4606			
encoder (char, word, ne_dic) / layer1	0.46951			
encoder (ne_dic, word) / layer1	0.79951			
encoder (word, ne_dic) / layer2	0.8161	0.8167238866		
encoder (word, ne_dic) / layer2	4: 0.919169(0.757045130489) / 0.579825 (0: 0.805011647674)			
encoder (word, ne_dic) / layer6	0: 0.779881878691, 1: 0.544266683664, 2: 0.684140860678			
encoder (word, ne_dic) / layer3				
encoder (word, ne_dic) / layer3 / shuffle	0.82765			
encoder (word, ne_dic_15) / layer3 / shuffle				
encoder (word, ne_dic_15) / layer8 / shuffle	0			
encoder (word, ne_dic_15) / layer4 / shuffle	0: 0.841120589376			
baseline + trans / layer4 / shuffle	FALSE			
encoder (word, ne_dic_15) / layer3 / shuffle				
encoder (word, ne_dic_15) / layer2 / shuffle				
encoder (word, ne_dic_15) / layer1 / shuffle				
embedding / encoder (word, ne_dic_15) / layer1 / shuffle				
embedding/ CNN / encoder (word, ne_dic_15) / layer1 / shuffle				
decoder (word(e) ne_dic(d)) / layer3	75에서 하강함			
decoder (word(d) ne_dic(e)) / layer3	56에서 하강함			

아쉬웠던 점들

- 학습데이터 분석의 미비
 - 학습데이터에 문제가 있지 않았나 의심을 해 봤어야 함
 - 결국 **base** 모델이 **baseline**을 넘지 못한 이유가 같은 원인이 아니었을까 생각 됨
- Attention, Transformer 모델에 심화부족
 - Self-Attention 계열의 모델들을 좀더 테스트 해보고 계선 해 봤으면 좋았을 텐데 하는 생각이 듬
- 원리적인 접근의 부족
 - 결과를 내는데 치중하지 않고 다른 모델을 시도해 보고 분석 해 봤으면 어떨까 하는 아쉬움

감사합니다

- 모두의연구소 자연어 처리반
- 도전팀 (명재님, 주성님, 성곤님)