Character-Aware Neural Language Models

김정원

Abstract

- Neural Network를 이용한 Language Model 제작
  - ConvNet, Highway네트워크, LSTM을 조합한 모델
- Word-level, 또는 형태소(morpheme)-level 단위를 사용하는 대신 Character-level을 통해 학습
  - 형태소가 많은 언어에 유리
- SOTA대비 성능의 개선은 없으나 60% 적은 parameter 사용

Introduction

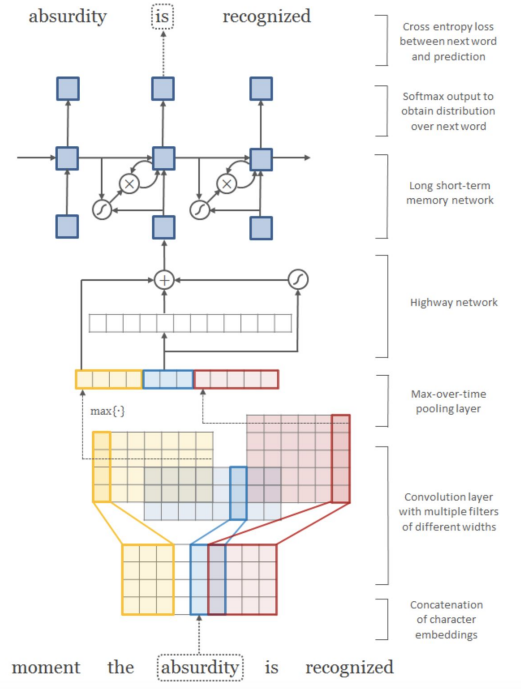
- 기존에는 n-th order Markov를 통한 추정이 많이 사용됨
  - n-gram의 확률을 계산하는 모델
  - 상대적으로 빈도수가 현저히 적은 n-gram에 대해서는 예측성능이 매우 저조
  - 이는 data sparsity의 문제때문
  - 이러한 기존 모델의 단점을 개선하기 위해 Neural Language Model이 사용됨
- Character-level CNN을 이용하여 subword 정보를 감안할 수 있는 Language Model을 만들고자 하는 것이 이번 논문의 목표

Introduction

- PTB(Penn Treebank) 데이터셋을 이용하여 당시 SOTA와 동일한 성능을 냄
    - 차이점은 60% 더 적은 parameter를 사용했다는 점
- 형태소가 많은 언어들에 대해 여러 baseline 성능을 넘김
    - 아라비아어, 체코어, 프랑스어, 독일어, 스페인어, 러시아어가 이에 해당

해당 모델은 https://github.com/yoonkim/lstm-char-cnn 에서 확인 가능

Model



- Input
- Concatenation
- Convolution Layer
- Max-over-pooling
- Highway network
- LSTM(Long short-term memory) Network
- Softmax output & Cross-entropy loss

Model

1. 각 글자별로 Embedding을 진행
    - Lookup Layer를 통해 글자 임베딩을 거치고 행렬 형태로 출력을 냄
2. ConvNet 연산은 임베딩 출력을 그대로 입력으로 받아 진행
3. Feature중 가장 중요한 요소만 남기기 위해 Max-over-time pooling
    - 여기서의 출력은 Highway Network의 입력으로 사용됨
4. Highway network의 출력은 LSTM 레이어의 입력으로 사용
5. 다음 단어에 대한 분포를 알아내기 위해 LSTM의 출력을 그대로 Affine Transformation을 적용 후, Softmax를 적용
    - 예측 단어의 분포와 실제 단어간의 차이를 줄이기 위해 Cross-entropy loss 사용

Model - LSTM

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^i \mathbf{x}_t + \mathbf{U}^i \mathbf{h}_{t-1} + \mathbf{b}^i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}^f \mathbf{x}_t + \mathbf{U}^f \mathbf{h}_{t-1} + \mathbf{b}^f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}^o \mathbf{x}_t + \mathbf{U}^o \mathbf{h}_{t-1} + \mathbf{b}^o) \\
\mathbf{g}_t &= \tanh(\mathbf{W}^g \mathbf{x}_t + \mathbf{U}^g \mathbf{h}_{t-1} + \mathbf{b}^g) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}
$$

- 기본 RNN모델에서는

  Gradient Vanishing 문제가 발생
  - LSTM을 통한 학습 진행
- Architecture
  - i_t: Input @ t
  - f_t: forget gate @ t
  - o_t: output gate @ t
  - g_t: memory gate @ t
  - c_t: cell state @ t
  - h_t: hidden state @ t

Model

$$NLL = -\sum_{t=1}^{T} log(p(w_t \mid w_{1:t-1}))$$

$$p(w_t \mid w_{1:t-1}) = fracexp(P^j h_t + q^j)\sum_{j'=1}^{V} exp(P^{j'} h_t + q^{j'})$$

V = 단어의 수, j = j-th column

- Loss: negative log-likelihood
  - Sequence에서 Word단위로 학습을 하기 위함
  - Truncated backpropagation through time을 이용하는 방식이 자주 쓰임

Model

Highway Network

- (1 - t)는 carry gate
- LSTM에서의 memory cell과 비슷한 역할
- 입력에서의 차원정보를 그대로 출력까지 '옮기는' 역할을 하기 때문

$$z = t \odot g(W_H y + b_H + (1 - t) \odot y)$$

- transform gate $t = \sigma(W_T y + b_T)$
- transform gate도 전체적인 model과 함께 학습됨

Experimental Setup

		Small	Large
CNN	d	15	15
	w	$[1, 2, 3, 4, 5, 6]$	$[1, 2, 3, 4, 5, 6, 7]$
	h	$[25 \cdot w]$	$[\min\{200, 50 \cdot w\}]$
	f	tanh	tanh
Highway	l	1	2
	g	ReLU	ReLU
LSTM	l	2	2
	m	300	650

- Stochastic Gradient Descent(SGD)
  - Learning rate decay
  - Perplexity > 1 -> LR/2
- Drop-out: 0.5
- Gradient norm
  - Gradient > 5 -> Norm = 5

Results

		DATA-S					
		CS	DE	ES	FR	RU	AR
Botha	KN-4	545	366	241	274	396	323
	MLBL	465	296	200	225	304	–
Small	Word	503	305	212	229	352	216
	Morph	414	278	197	216	290	230
	Char	401	260	182	189	278	196
Large	Word	493	286	200	222	357	172
	Morph	398	263	177	196	271	**148**
	Char	**371**	**239**	**165**	**184**	**261**	**148**

- 작은 데이터셋에서
  - 모든 실험 대상 언어에 대해 글자 단위의 입력이 결과가 좋았음
  - 예외적으로 아라비아어의 경우 형태소 단위와 글자 단위의 입력 사이의 성능 차이가 없음
- 모든 실험 대상 언어에 대해 논문에서 제안한 모델이 perplexity 점수가 더 좋게 나옴

Results

		DATA-L					
		CS	DE	ES	FR	RU	EN
Botha	KN-4	862	463	219	243	390	291
	MLBL	643	404	203	227	**300**	273
Small	Word	701	347	186	202	353	236
	Morph	615	331	189	209	331	233
	Char	**578**	**305**	**169**	**190**	313	**216**

- 큰 데이터셋의 경우
  - 형태소의 종류가 많은 언어를 대상
  - 모든 경우에 단어/형태소 단위의 입력보다 글자 단위의 입력이 좋은 성능을 보임
  - 때로는 단어단위의 입력과 형태소 단위의 입력에서 차이를 보이지 않았을 경우도 많음
- 러시아어를 제외한 모든 언어에 대해 기존의 모델보다 논문에서 제시한 모델의 성능이 더 좋게 나옴

Results

	In Vocabulary				
	while	*his*	*you*	*richard*	*trading*
LSTM-Word	*although*	*your*	*conservatives*	*jonathan*	*advertised*
	letting	*her*	*we*	*robert*	*advertising*
	though	*my*	*guys*	*neil*	*turnover*
	minute	*their*	*i*	*nancy*	*turnover*
LSTM-Char (before highway)	*chile*	*this*	*your*	*hard*	*heading*
	whole	*hhs*	*young*	*rich*	*training*
	meanwhile	*is*	*four*	*richer*	*reading*
	white	*has*	*youth*	*richter*	*leading*
LSTM-Char (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*
	whole	*this*	*your*	*gerard*	*training*
	though	*their*	*doug*	*edward*	*traded*
	nevertheless	*your*	*i*	*carl*	*trader*

Results

	LSTM-Char	
	Small	Large
No Highway Layers	100.3	84.6
One Highway Layer	92.3	79.7
Two Highway Layers	90.1	78.9
One MLP Layer	111.2	92.6

- Highway Layer가 많다고 성능이 잘 나오지는 않았음
- ConvNet Layer는 최대 2개가 제일 성능이 잘 나옴
- 데이터셋이 작은 경우에는 모델의 성능이 잘 나오지 않음
    - 데이터셋이 큰 경우에 성능이 나옴

Results

- Highway Network가 단어의 Semantic Meaning을 이해하는데 많은 영향을 끼침
- Highway Network를 여러개 쌓을때의 성능은 그리 좋지 않았음
    - 레이어 1개 추천

Conclusion

- Character-level input만을 이용한 모델이 좋은 성능을 보임
- 단어/형태소 단위의 입력에 비해 글자만 사용한 모델이 훨씬 좋은 성능을 보임
  - 단어가 표현하는 바를 글자 단위로 Composition할 수 있던 것이 성능의 이유로 보임
  - 이에 대해 Word2Vec을 Highway Network에 사용하고자 하는 것에 대한 연구 예정

References

- 참고자료: https://github.com/YBIGTA/DeepNLP-Study/wiki/Character-Aware-Neural-Language-Model(2016)
- 논문링크: https://arxiv.org/abs/1508.06615