Character-Aware

Neural Language Models

Deep NLP Lab

2019/03/20

Contents

Character-Aware Neural Language Models

Character-Aware Neural Language Models

Yoon Kim[†] **Yacine Jernite**[*] **David Sontag**[*] **Alexander M. Rush**[†]

[†]School of Engineering and Applied Sciences
Harvard University
{yoonkim,srush}@seas.harvard.edu

[*]Courant Institute of Mathematical Sciences
New York University
{jernite,dsontag}@cs.nyu.edu

Abstract

We describe a simple neural language model that relies only on character-level inputs. Predictions are still made at the word-level. Our model employs a convolutional neural network (CNN) and a highway network over characters, whose output is given to a long short-term memory (LSTM) recurrent neural network language model (RNN-LM). On the English Penn Treebank the model is on par with the existing state-of-the-art despite having 60% fewer parameters. On languages with rich morphology (Arabic, Czech, French, German, Spanish, Russian), the model outperforms word-level/morpheme-level LSTM baselines, again with fewer parameters. The results suggest that on many languages, character inputs are sufficient for language modeling. Analysis of word representations obtained from the character composition part of the model reveals that the model is able to encode, from characters only, both semantic and orthographic information.

While NLMs have been shown to outperform count-based n-gram language models (Mikolov et al. 2011), they are blind to subword information (e.g. morphemes). For example, they do not know, a priori, that *eventful*, *eventfully*, *uneventful*, and *uneventfully* should have structurally related embeddings in the vector space. Embeddings of rare words can thus be poorly estimated, leading to high perplexities for rare words (and words surrounding them). This is especially problematic in morphologically rich languages with long-tailed frequency distributions or domains with dynamic vocabularies (e.g. social media).

In this work, we propose a language model that leverages subword information through a character-level convolutional neural network (CNN), whose output is used as an input to a recurrent neural network language model (RNN-LM). Unlike previous works that utilize subword information via morphemes (Botha and Blunsom 2014; Luong, Socher, and Manning 2013), our model does not require morphological tagging as a pre-processing step. And, unlike

Sequence to sequence Learning: Abstract

- 우리 Paper는 Character-level input에만 의존하는 직관적인 Neural Language Model 제안

  → Prediction은 여전히 word-level에서 이루어짐

- Character input을 CNN과 highway network에 통과시키고, 그 output을 LSTM LM의 input으로 사용
- **영어**: Penn Treebank (PTB)에서 60% 더 적은 parameter들을 가지고 SOTA model과 동등한 성능을 보임
- **형태소 풍부 언어**: 더 적은 parameter들을 가지고 word-level/morpheme-level LSTM LM의 baseline 능가

  → (아랍어, 체코어, 불어, 독일어, 스페인어, 러시아어)

- 위 결과는 많은 언어에 있어, character-level input이 Language Modeling에 충분하다는 것을 시사

  "*Character 조합으로 이루어진 단어 표현만으로도 semantic, orthographic한 정보를 인코딩할 수 있다 !* "

Sequence to sequence Learning: Introduction (background)

- Word embedding을 사용하는 NLM은 형태소 같은 **subword information**을 찾지 못한다는 단점 지님

  e.g.) *eventful, eventfully, un-eventful, uneventfully* 과 같은 단어들이 vector 공간에서 유사한 embedding을 보여야 하는데 그러지 못하고, 이 중 드물게 등장하는 단어의 embedding은 잘 예측되지 않아 perplexity가 높아짐

  → eventful이 드물게 등장해도, eventfully가 많이 나왔다면 그와 비슷하게 embedding을 하면 되는데 그러지 못함

  → 따라서 eventful을 잘 예측하지 못하고 (data sparsity), 그와 유사한 의미의 단어들의 예측 확률 역시 줄어들게 됨

- **그리고 이는 <u>형태소가 풍부한 언어에 있어 큰 장애물이 될 수 있음</u>**

Sequence to sequence Learning: Introduction

- 따라서, 우리는 **char-level CNN을 통해 subword information을 얻어 활용하는 모델을 제안** !
- 형태소 분석 전처리를 통해 subword information을 활용한 이전의 연구들과는 다르게,

  우리 모델은 **형태소 태깅**을 전처리 작업으로 필요하지 않음
- 또한, char-level model의 output feature와 word-embedding을 combine하는 요즘 연구들과는 다르게,

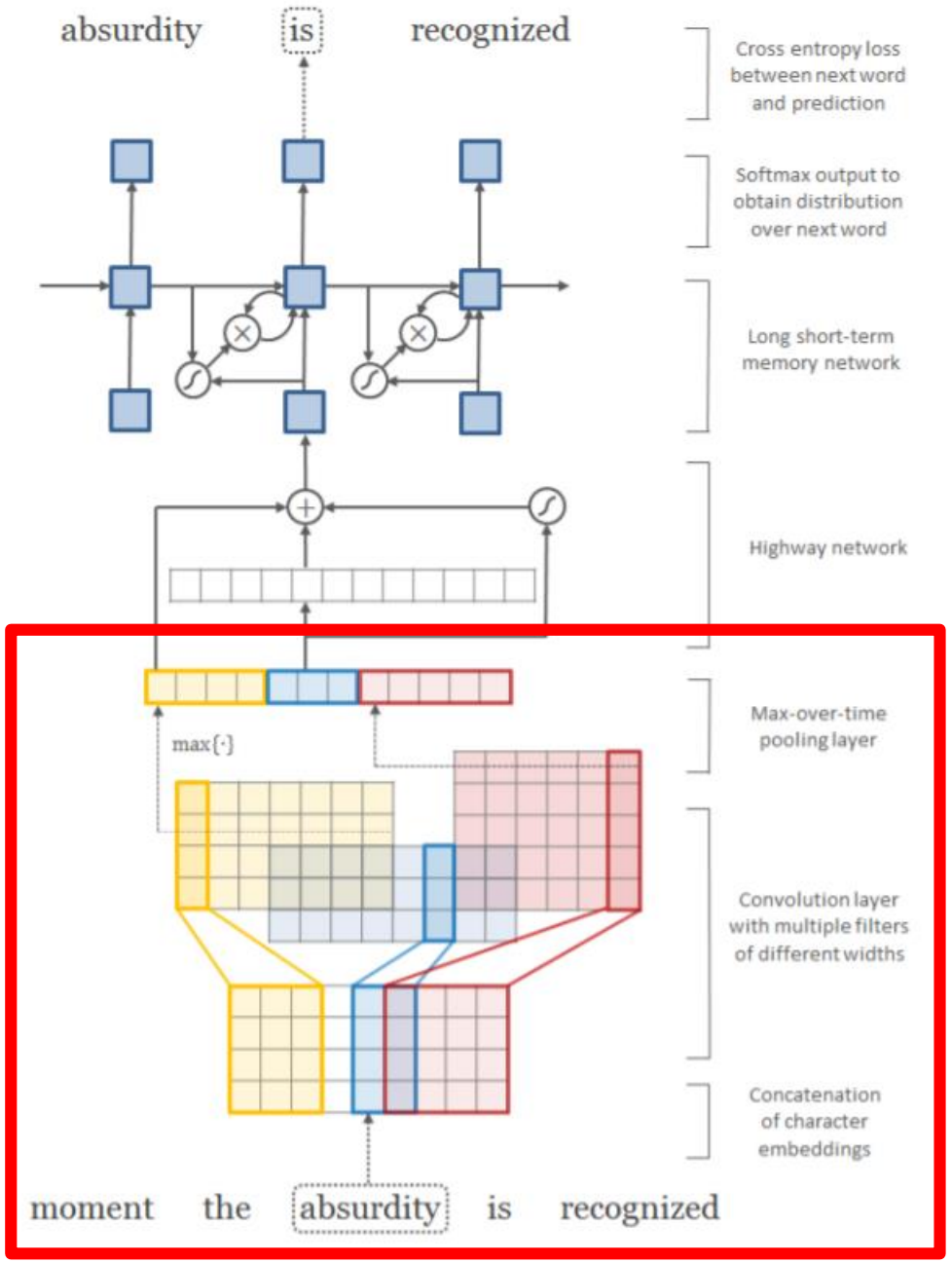
  우리 모델은 **word embedding**을 input layer에서 전혀 사용하지 않음
- NLM의 대부분의 parameter는 word embedding에서 발생

  → 우리 모델은 word embedding 사용 않기에 parameter 수가 이전 NLM 모델들에 비해 현저히 적고,

  이는 model size가 중요한 환경에서 우리 model이 매력적이게 만듦 (e.g. cell phone)

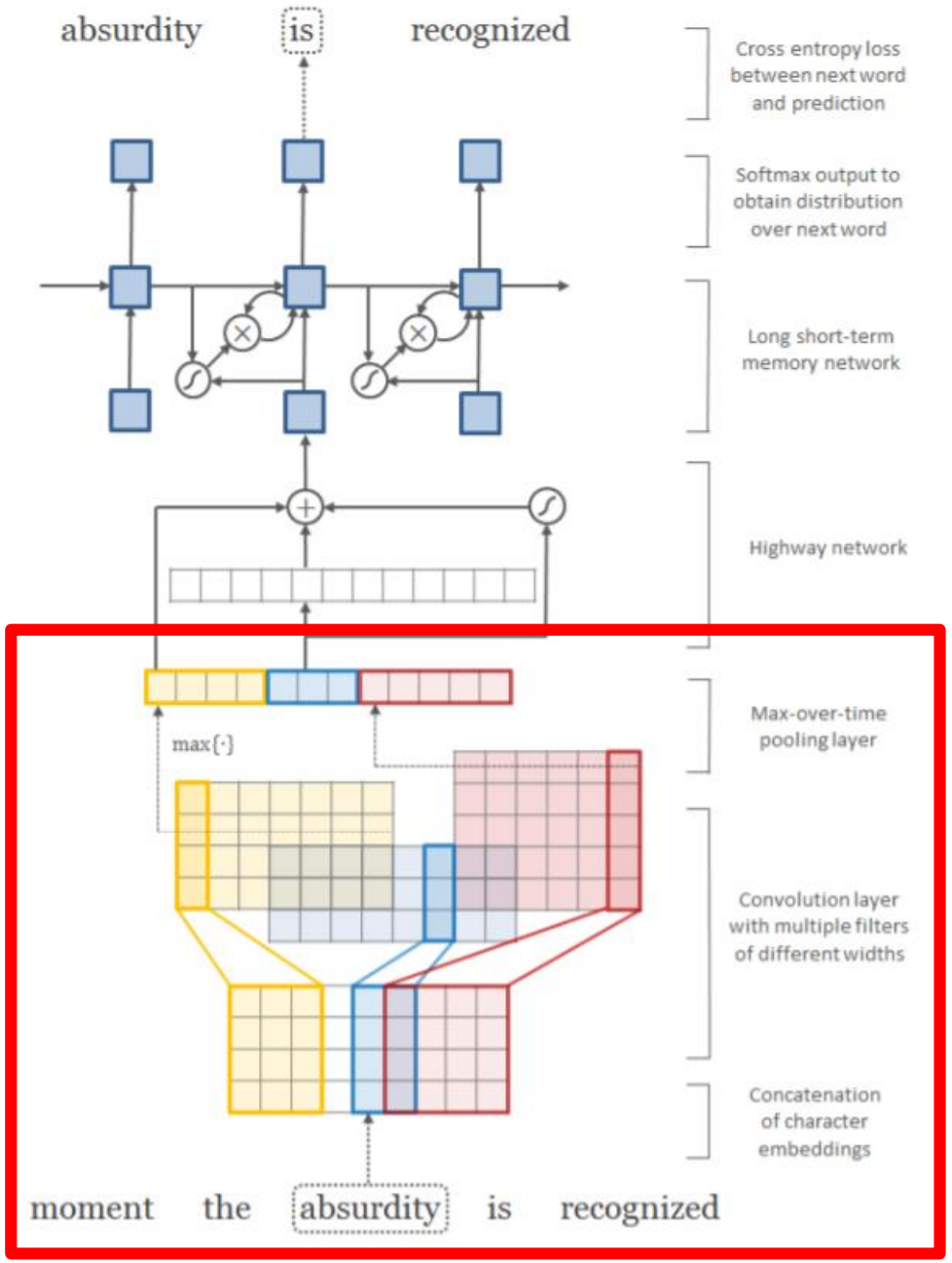
Sequence to sequence Learning: Model

- time "**t**" 때 model input은 char-level CNN's output
- 1) 단어를 구성하는 **각각의** character embedding을 concatenate하여, Matrix character embedding 구성
- 2) Matrix character embedding에 **다양한 크기**의 filter를 적용하여 다양한 feature map들을 획득
- 3) 마지막으로 각각의 filter를 통해 생성된 feature map에 Max pooling 적용해 가장 중요한 feature 값 추출



Sequence to sequence Learning: Model

- **Filter는 본질적으로 character의 n-gram을 뽑아 냄**
- 이 때, n-gram의 n 크기는 filter 크기와 같음
  - 2-size filter == bigram, 3-size filter == trigram …
- 우리 모델의 charCNN은 단어 "k"의 feature vector를 많이 추출하기 위해 **<u>다양한 크기</u>**의 filter를 **<u>여러 개</u> 사용**함
- 많은 NLP application에서 보편적으로 사용되는 filter 개수는 **[100, 1000] 개**



Sequence to sequence Learning: Model

- Max-pooling을 통해 나온 output “y”를

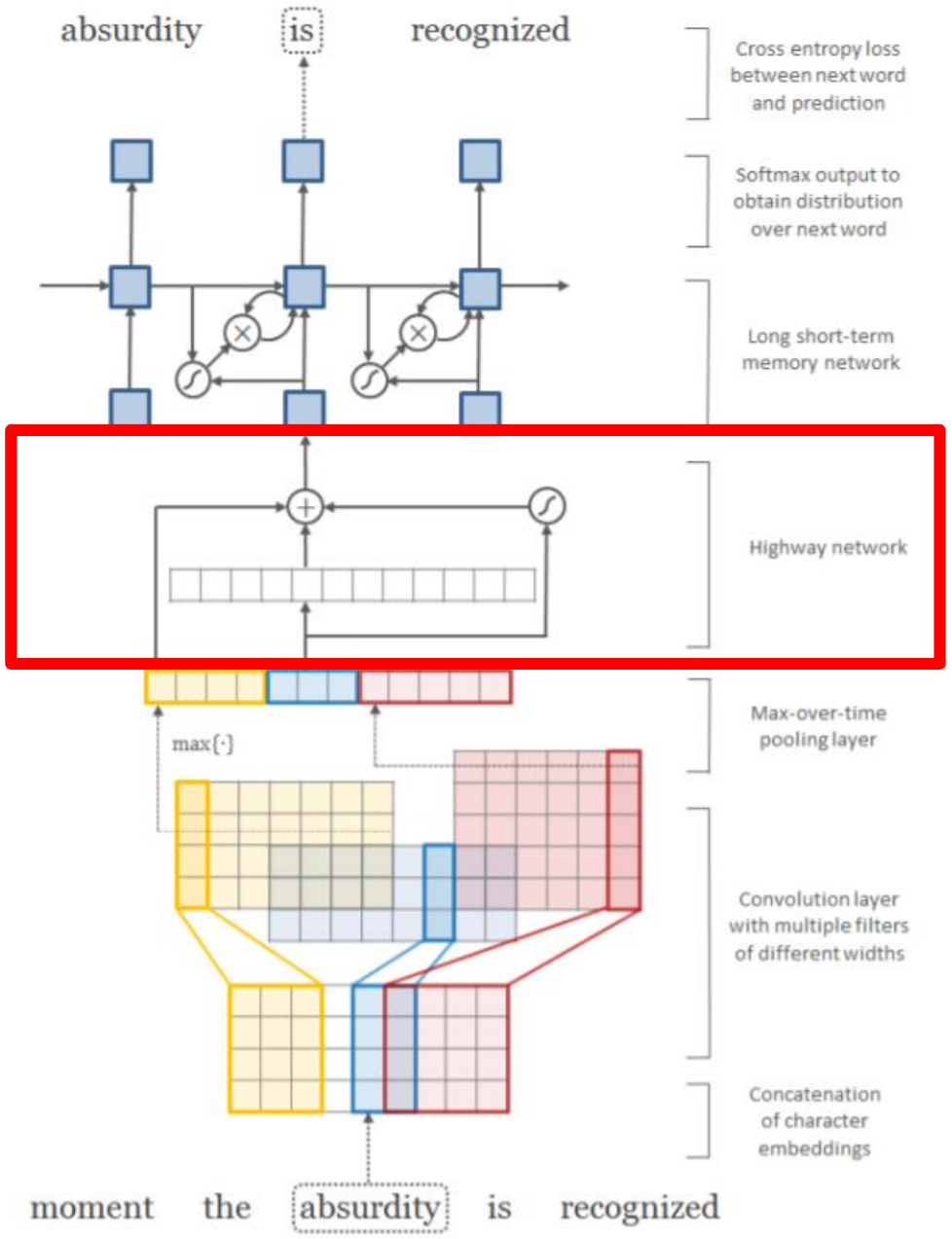
  highway network를 거쳐 LSTM으로 보내어 성능 개선

  → highway layer는 LSTM memory cell과 유사하게

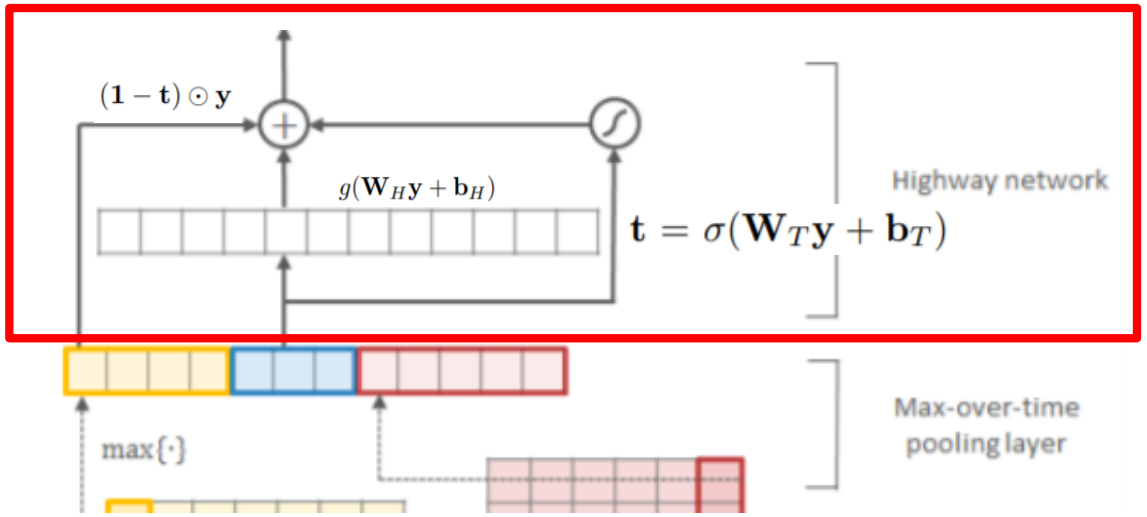
  input의 몇몇 값들을 다음 계층으로 direct하게 보내줌

- *+) Highway network는 모델의 depth를 늘려주고,*

  네트워크를 최적화 하는 역할을 수행



Sequence to sequence Learning: Highway network



one layer of a highway network does the following:

$$\mathbf{z} = \mathbf{t} \odot g(\mathbf{W}_H\mathbf{y} + \mathbf{b}_H) + (\mathbf{1} - \mathbf{t}) \odot \mathbf{y} \qquad (8)$$

where g is a nonlinearity, $\mathbf{t} = \sigma(\mathbf{W}_T\mathbf{y} + \mathbf{b}_T)$ is called the *transform* gate, and $(\mathbf{1}-\mathbf{t})$ is called the *carry* gate. Similar to the memory cells in LSTM networks, highway layers allow for training of deep networks by adaptively *carrying* some dimensions of the input directly to the output.[5] By construction the dimensions of $\mathbf{y}$ and $\mathbf{z}$ have to match, and hence $\mathbf{W}_T$ and $\mathbf{W}_H$ are square matrices.

- 수식에서의 **t**는 sigmoid를 적용하는 transform gate: 즉, network를 통해 변경하는 값들의 비율
- **(1-t)**는 carry gate: network를 통해 변경하지 않고, 다음 계층으로 바로 전달하고자 하는 값들의 비율
- t 수식의 output은 0~1 사이 벡터 값이므로, 해당 값 만큼만 transform하고 나머지는 directly carry
- 즉, highway network는 **gradient가 잘 흐를 수 있도록 shortcut(carry gate)을 부여해줌 !**

Sequence to sequence Learning: Experimental Setup

- **성능척도**: LM의 성능을 판단하는 standard 인 **Perplexity(PPL)** : $PPL = \exp\left(\frac{NLL}{T}\right)$
- 모델을 **다양한 Dataset size**를 가진 언어들에 대해 실험
- PTB를 사용해 **하이퍼 파라미터 설정, 모델 재점검, highway network를 붙이고 떼는 등**의 실험 수행

  이 때, standard training (0-20), validation (21-22), test (23-24)의 split을 나누어 train을 진행하였음
- 이후, PTB를 통해 최적으로 맞춰진 하이퍼 파라미터를 가지고, 모델을 형태소가 풍부한 다양한 언어들에 Train !

* *Non-Arabic data* comes from the 2013 ACL MT Workshop, *Arabic data* comes from the News-Commentary corpus

Sequence to sequence Learning: Experimental Setup

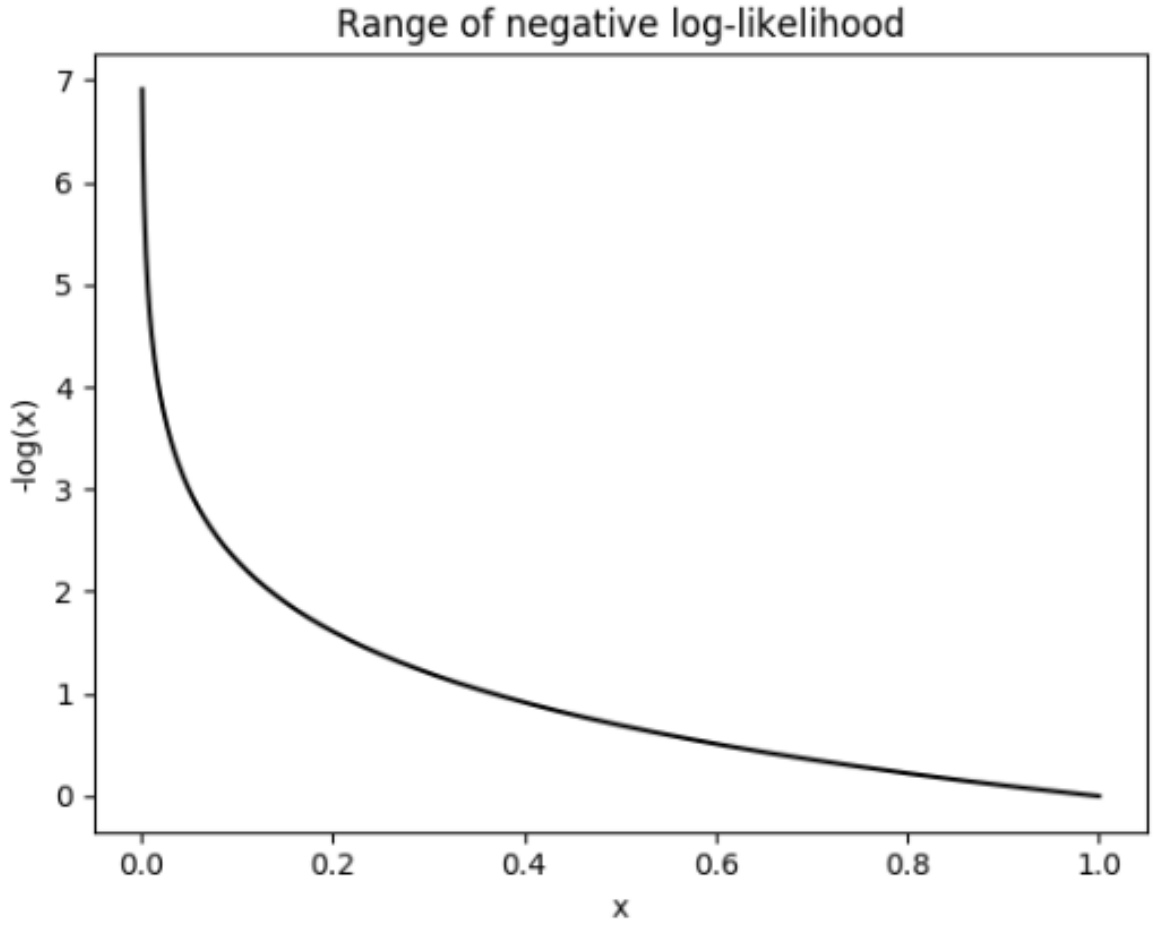


Figure: *The loss function reaches infinity when input is 0, and reaches 0 when input is 1.*

$$PPL = \exp\left(\frac{NLL}{T}\right)$$

- NLL은 모델의 예측이 틀릴 때, cost를 무한대에 가깝게 주고, 예측이 맞을 때 0의 cost 부여
- 따라서 Testcase에 대해 Language modeling의 Negative Log Likelihood를 줄이는 것이 목표

Sequence to sequence Learning: Experimental Setup

- 언어 당 1m 개의 token을 가지는 **small dataset**과, 그 이상의 **large dataset**. 두 dataset에 대하여 train 진행
- Dataset에서 singleton words만 <UNK>로 대체하였기 때문에 almost full Vocab을 효과적으로 활용하였음
- +) 우리의 character-level model은 OOV(Out-Of-Vocabulary) token의 표면적 형태를 활용할 수 있음

  e.g.) **"well-organized"**라는 단어가 Vocab에 없을 때, 각 character의 embedding을 combine해서 해당 단어의 embedding vector를 생성 가능해지고, 이에 따라 대략적인 word representation의 생성도 가능해짐
- 그러나, 이전 연구들과의 확실한 비교 위해 이러한 장점을 채택하지 않고, <UNK> 그대로 사용

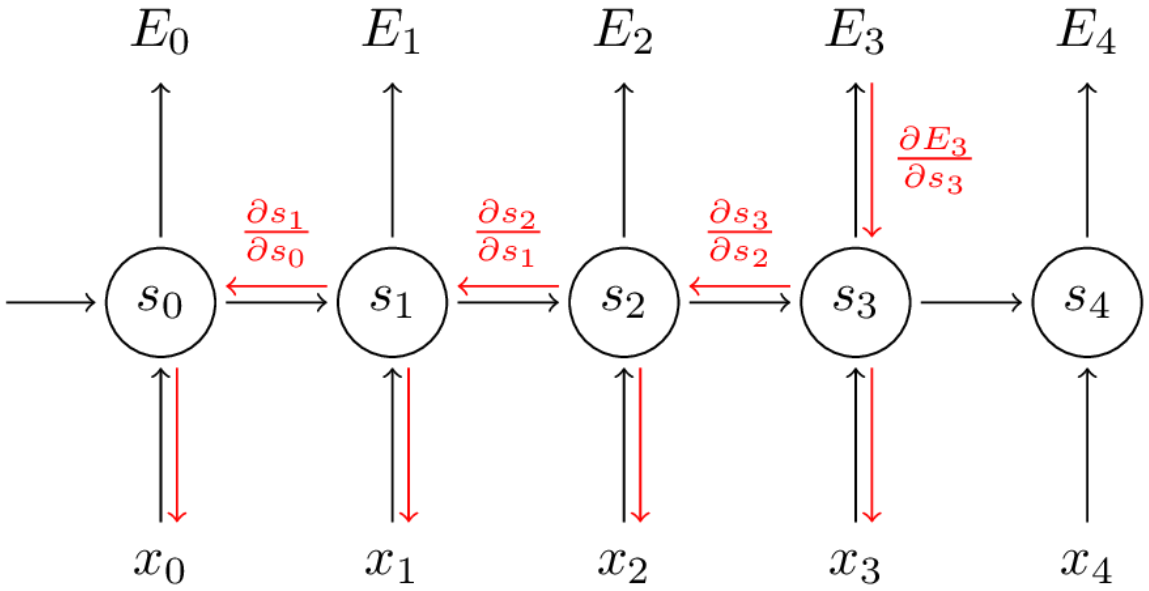
Sequence to sequence Learning: Optimization

- 우리 모델은 **“ Truncated BPTT ”**에 의해 train 되었으며, 35 time steps 만큼 역전파 수행
- **SGD**를 사용 : Learning rate를 **1.0**으로 초기화하고, 한 번의 epoch 이후 validation set을 통해

  검증해보았을 때 perplexity가 1.0 이상 감소하지 않을 경우 Learning rate를 **½**로 줄임
- DATA-S에는 20 size의 batch 사용, DATA-L에는 100 size의 batch 사용
- **Gradient는 매 batch마다 총합의 평균을 사용 →** 특정 data의 큰 gradient를 평균을 통해 제지하기 위해 ! (정규화)
- Non-Arabic 언어에는 25 epochs 적용, Arabic 에는 30 epochs 적용하여

  validation set에서 가장 좋은 성능을 보인 모델을 선정

Sequence to sequence Learning: Backpropagation Through Time



- RNN의 **t' hidden state는 이전의 t-1' hidden state에 영향을 받기 때문에**, 한 time step (e.g. E3)의 역전파를 위해서는 이전의 time step들 (e.g. E0 - E2)까지 순차적으로 역전파를 수행하여야 함 → " **BPTT** "
- 이 때, 역전파 출발점이 너무 멀다면 E0까지 수행해야 할 역전파의 Time step 수가 너무 많아 연산량 증가 야기
- 이러한 문제를 해결하기 위해 ***정해진 Time step 수*** **만큼만 역전파를 수행하는 "** Truncated **BPTT "를 주로 사용**

Sequence to sequence Learning: Optimization

- 모델의 초기 parameter들은 **[-0.05, 0.05]의 연속 균등 분포**로 초기화
- 정규화를 위해 LSTM의 input-to-hidden layer, hidden-to-output softmax layer에 0.5 확률의 **dropout** 적용
- ***Gradient Clipping****: Gradient의 norm을 5 이하로 제한하여, gradient의 L2 norm 이 5를 초과할 경우,*

  이를 재정규화해 update 이전에 ||·|| *(Gradient norm)가 5가 되게 함*

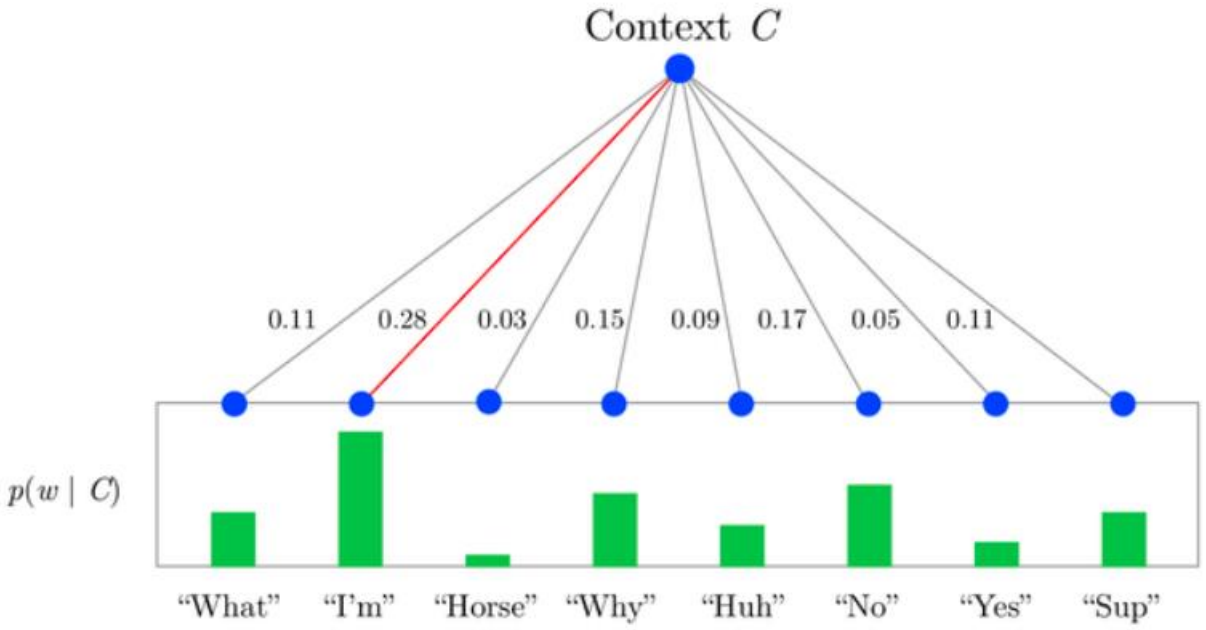
$$L_2 = \sqrt{\sum_i^n x_i^2} = \sqrt{x_1^2 + x_2^2 + x_3^2 + \ldots + x_n^2}$$

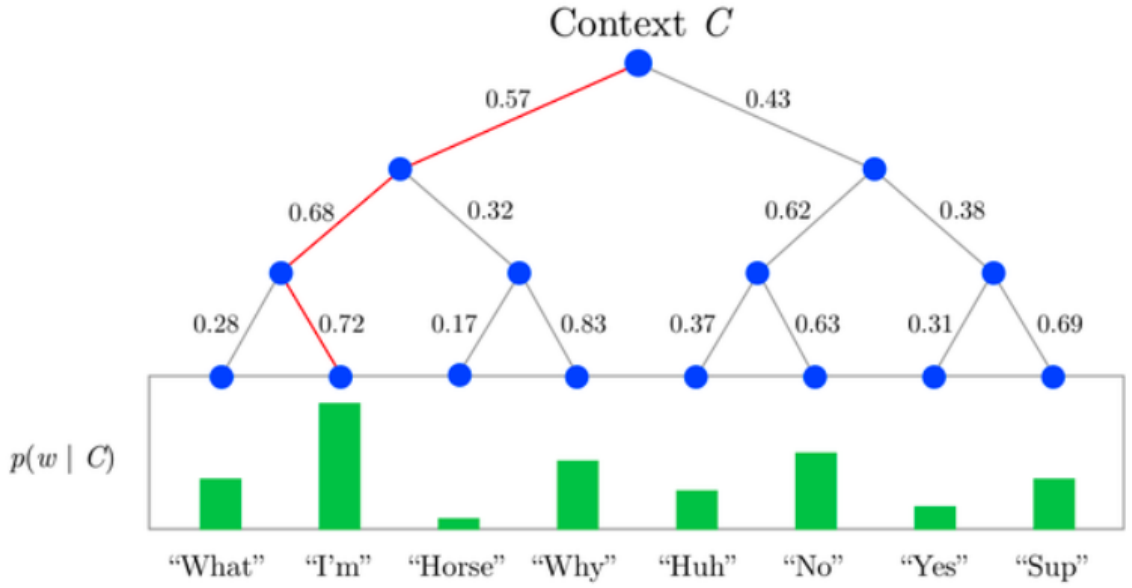
  → gradient가 너무 커져, **Gradient exploding 문제**가 발생하는 것을 막기 위한 정규화의 일환

- 마지막으로, DATA-L에서의 training을 가속화하기 위해 " **hierarchical softmax** "를 적용

  → hierarchical softmax는 **매우 큰 Vocab**을 가지는 LM을 훈련시킬 때 보편적으로 적용되는 전략

Sequence to sequence Learning: hierarchical softmax





To evaluate the probability of a given word, take the product of the probabilities of each edge on the path to that node:

$$P(\text{I'm}|C) = 0.57 * 0.68 * 0.72 = 0.28$$

- Softmax function은 **Tree**와 같은 형태: Context vector "C"를 **root**로 하고, Vocab 단어들의 확률이 **leaves**
- But, Vocab size가 너무 크면 연산량 역시 지나치게 커지고, Context C 하나의 softmax를 위해 *O(n)*의 연산 수행
- 이를 개선하기 위해, **다층 Tree 활용: 주어진 단어의 확률을 계산하기 위해 edge path의 확률을 계산 !** *O(log n)*

Sequence to sequence Learning: Results (English Penn Treebank)

		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

Table 2: Architecture of the small and large models. d = dimensionality of character embeddings; w = filter widths; h = number of filter matrices, as a function of filter width (so the large model has filters of width $[1, 2, 3, 4, 5, 6, 7]$ of size $[50, 100, 150, 200, 200, 200, 200]$ for a total of 1100 filters); f, g = nonlinearity functions; l = number of layers; m = number of hidden units.

- **모델의 크기와 성능의 상관관계**를 살펴 보기 위해,

  2개 버전의 모델을 학습

- 왼쪽의 Small이 작은 크기의 model,

  오른쪽의 Large가 큰 크기의 model

Sequence to sequence Learning: Results (English Penn Treebank)

- 또 다른 baseline 추가 위해, **word embedding** 활용한 모델들도 학습 (200/650 hidden layers, respectively)
- 표에서 보다시피, 우리의 large model은 SOTA와 동등한 성능을 60% 더 적은 parameter으로 달성
- Small model 역시 비슷한 크기의 다른 NLM 모델들에 비해 더 좋은 성능을 보임

  → *OOV words을 활용하지 않고도 얻은 성과 !*

	PPL	Size
LSTM-Word-Small	97.6	5 m
LSTM-Char-Small	92.3	5 m
LSTM-Word-Large	85.4	20 m
LSTM-Char-Large	78.9	19 m
KN-5 (Mikolov et al. 2012)	141.2	2 m
RNN† (Mikolov et al. 2012)	124.7	6 m
RNN-LDA† (Mikolov et al. 2012)	113.7	7 m
genCNN† (Wang et al. 2015)	116.4	8 m
FOFE-FNNLM† (Zhang et al. 2015)	108.0	6 m
Deep RNN (Pascanu et al. 2013)	107.5	6 m
Sum-Prod Net† (Cheng et al. 2014)	100.0	5 m
LSTM-1† (Zaremba et al. 2014)	82.7	20 m
LSTM-2† (Zaremba et al. 2014)	78.4	52 m

Sequence to sequence Learning: Results (Other Languages)

- PTB에서 입증된 model의 성능도 주목할 만하지만, 영어는 형태소적 관점에서 봤을 때 간단한 언어
- **우리의 main contribution은 영어가 아닌 형태소가 풍부한 언어에 있다 !**
- 우리 모델을 **morphological log-bilinear (MLBL)** 라는 모델과 비교

  해당 모델 역시 *morpheme embedding*을 통해 **subword information**을 고려한 모델
- 해당 모델과 비교하기 위해, morpheme embedding을 input으로 사용하는 LSTM NLM 모델도 생성
- +) 또 다른 baseline으로 삼기 위해 word-level LSTM model 까지 생성 !

Sequence to sequence Learning: Results (Other Languages)

		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**

		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**

- In DATA-S, char-level이 모든 다른 모델 (word/morpheme level, MLBL) 보다 좋은 성능을 보임
- *cf) Morpheme level이 word level 보다 좋은 성능을 보였지만, morpheme model은 "**word embedding + morpheme embeddings**"를 input으로 사용하기 때문에 훨씬 더 많은 parameter가 사용된다는 점이 고려되어야 함*

Sequence to sequence Learning: Results (Other Languages)

		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**

		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**

- 메모리 제약으로 DATA-L에 대해서는 **small model**만 train 시킬 수 있었는데, 스페인어, 불어, 영어에서 word/morpheme-level 간 차이가 거의 발견되지 않았다는 점이 흥미로웠음
- +) char-level model이 다른 모델들 모두 outperform
- 우리는 **모든 언어에 대해 같은 아키텍쳐**를 사용했고, 특정 언어를 위한 하이퍼 파라미터 튜닝은 수행하지 않음

Sequence to sequence Learning: Discussion

<1. Learned Word Representations>

	In Vocabulary					Out-of-Vocabulary		
	while	*his*	*you*	*richard*	*trading*	*computer-aided*	*misinformed*	*looooook*
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*	*computer-guided*	*informed*	*look*
	whole	*hhs*	*young*	*rich*	*training*	*computerized*	*performed*	*cook*
	meanwhile	*is*	*four*	*richer*	*reading*	*disk-drive*	*transformed*	*looks*
	white	*has*	*youth*	*richter*	*leading*	*computer*	*inform*	*shook*
LSTM-Char (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*	*computer-guided*	*informed*	*look*
	whole	*this*	*your*	*gerard*	*training*	*computer-driven*	*performed*	*looks*
	though	*their*	*doug*	*edward*	*traded*	*computerized*	*outperformed*	*looked*
	nevertheless	*your*	*i*	*carl*	*trader*	*computer*	*transformed*	*looking*

Highway layer를 적용하기 이전에는 representation이 단순히 단어의 **표면적 형태**에 의존하는 것으로 보임

e.g.) you에 근접한 이웃: your, young, four, youth. **즉, you와 편집거리가 가까운 단어들 !**

Sequence to sequence Learning: Discussion

<1. Learned Word Representations>

	In Vocabulary					Out-of-Vocabulary		
	while	*his*	*you*	*richard*	*trading*	*computer-aided*	*misinformed*	*looooook*
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*	*computer-guided*	*informed*	*look*
	whole	*hhs*	*young*	*rich*	*training*	*computerized*	*performed*	*cook*
	meanwhile	*is*	*four*	*richer*	*reading*	*disk-drive*	*transformed*	*looks*
	white	*has*	*youth*	*richter*	*leading*	*computer*	*inform*	*shook*
LSTM-Char (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*	*computer-guided*	*informed*	*look*
	whole	*this*	*your*	*gerard*	*training*	*computer-driven*	*performed*	*looks*
	though	*their*	*doug*	*edward*	*traded*	*computerized*	*outperformed*	*looked*
	nevertheless	*your*	*i*	*carl*	*trader*	*computer*	*transformed*	*looking*

Highway layer 적용 후에는 철자만으로는 얻을 수 없는 **semantic feature**까지도 잡아낼 수 있게 됨: *you* – *we*

+) *while*과 *though*의 편집거리는 굉장히 멀지만 두 단어를 semantic하게 유사한 단어로 잘 잡아냄 !

Sequence to sequence Learning: Discussion

<1. Learned Word Representations>

	In Vocabulary					Out-of-Vocabulary		
	while	*his*	*you*	*richard*	*trading*	*computer-aided*	*misinformed*	*looooook*
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*	*computer-guided*	*informed*	*look*
	whole	*hhs*	*young*	*rich*	*training*	*computerized*	*performed*	*cook*
	meanwhile	*is*	*four*	*richer*	*reading*	*disk-drive*	*transformed*	*looks*
	white	*has*	*youth*	*richter*	*leading*	*computer*	*inform*	*shook*
LSTM-Char (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*	*computer-guided*	*informed*	*look*
	whole	*this*	*your*	*gerard*	*training*	*computer-driven*	*performed*	*looks*
	though	*their*	*doug*	*edward*	*traded*	*computerized*	*outperformed*	*looked*
	nevertheless	*your*	*i*	*carl*	*trader*	*computer*	*transformed*	*looking*

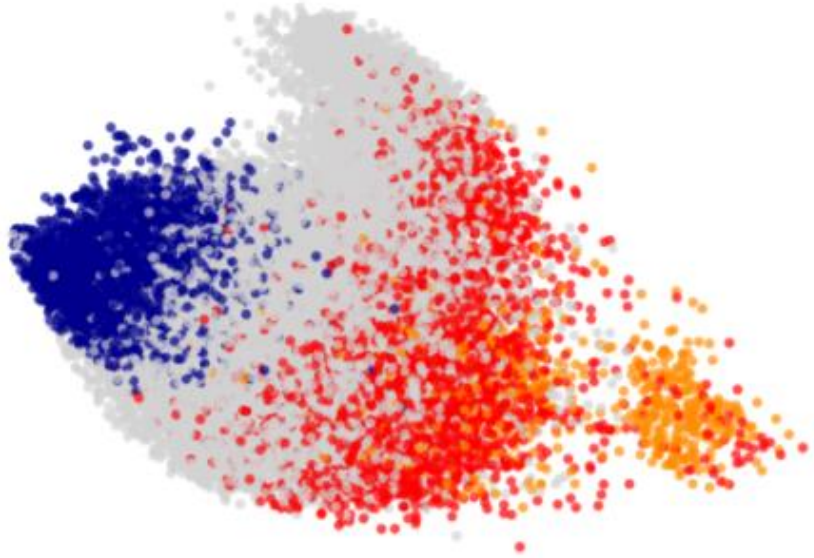
- 그러나, 우리 모델은 뚜렷한 실수들도 생성해내며 접근 방법의 한계를 보이기도 함: *his – hhs*

- 학습을 통해 형성한 OOV 단어들의 word representation도 생성해냈으며,

  심지어 틀린 스펠링까지 교정 해내며 **텍스트 정규화** 어플리케이션으로 활용될 수 있는 잠재력까지 내비침

Sequence to sequence Learning: Discussion

<2. Learned Character N-gram Representations>



- CharCNN의 각 filter는 특정 char n-gram을 탐지하도록 훈련
- 이에 따라 우리의 초기 기대는 *각 filter가 서로 다른 형태소를 활성화하고,*

  활성화된 형태소들이 모여 semantic representation을 생성하지 않을까? 하는 것이었음
- 그러나, 학습된 char n-gram을 살펴 보니, n-gram들이 **대체로 valid morpheme이 아니었음**
- " Model이 어떠한 char n-gram을 학습하는지 알아보기 위해, 모든 char n-gram을 PCA로 찍어 보자 ! "

  결과: 모델은 *접두사 (red)*, *접미사 (blue)*, *하이픈 단어 (orange)* 그리고 기타 (grey)를 분류하는 법을 학습하고 있었음

  → 추정컨대 아마 위 정보들이 단어의 POS Tagging에 **중요한 역할**을 수행하기 때문일 것

Sequence to sequence Learning: Discussion

<3. Highway Layers>

	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 network의 유무에 따른 모델의 성능 개선**을 조사하기 위한 연구도 수행
- 1) Highway layer를 제거하고 train 했을 때, 성능이 크게 감소

  ** 이는, Model size의 감소로 인한 성능 감소일 수도 있었기에 CharCNN output을 MLP에 넘겨주는 방식도 실험*

  → 이 역시, 좋은 성능을 보이지 않았음
- 따라서 우리는 **highway network과 CNN의 조합이 효과적**이라는 가설을 설정하게 됨

Sequence to sequence Learning: Discussion

<3. Highway Layers>

	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 network가 좋은 이유는 개별 filter에서 나온 local feature들을 **적절하게 조합**해주기 때문
- 2) **1~2개의 highway layer를 놓는 것이 좋으며**, 이 이상의 개수는 성능 개선 보이지 않음 *(dataset 크기 때문일수도..)*
- Max-pooling 이전에 여러 개의 Convolution layer를 쌓는 것은 큰 도움이 되지 않았음
- Word embedding을 input으로 사용하는 model에 highway network를 추가하는 것은 큰 성능 개선 보이지 않음

  → *"CharCNN + highway network의 조합을 통해 NLP에 또 다른 발전을 이룰 수도 있지 않을까? "*

Sequence to sequence Learning: Discussion

<4. Effect of Corpus/Vocab Sizes>

		$\lvert\mathcal{V}\rvert$			
		10 k	25 k	50 k	100 k
T	1 m	17%	16%	21%	–
	5 m	8%	14%	16%	21%
	10 m	9%	9%	12%	15%
	25 m	9%	8%	9%	10%

- **Training corpus/Vocab 크기에 따른 성능 개선 효과**를 알아보기 위해, DATA-L에서 독일어 dataset을 가져와 크기를 달리 적용하며 word-level model에 비해 char-level model의 perplexity가 얼마나 감소하는지 기록

  → Dataset을 줄이기 위해 이전 크기에서 가장 많이 사용된 단어들을 <UNK>로 대체해가는 방식을 채택

- 표는 corpus 크기가 커질수록 word-level model에서 char-level model로의 perplexity 감소가 작아진다는 것을 보이고 있지만, **그럼에도 모든 시나리오에서 char-level model이 word-level model을 압도**하는 것을 확인하였음

Sequence to sequence Learning: Further Observations

- 1) CharCNN output + word embedding의 “combined word representation” 사용이 더 안 좋은 성능 보임

→ 이는 굉장히 흥미로운 결과인데 POS Tagging과 NER에서는 위 방법이 성능 개선을 보였기 때문

→ 이는 몇몇 NLP tasks에서는 character input만으로도 충분하고, Word embedding은 **과잉의 정보**라는 점을 시사

- 2) 우리 모델은 character에 추가적인 convolution 연산을 수행하기 때문에 word-level 모델보다 느릴 수 밖에 없음

  ** word-level model은 input layer에서 간단하게 word를 lookup만 하면 되기 때문*

→ 그러나, ***Vocab에 존재하는 모든 단어들의 CharCNN output*** 을 미리 가지고 있으면 Scoring 때는 word-level과 비슷한 속도 낼 수 있음. 즉, **수행 시간의 감소**와 **model size의 증가로 인한 memory 증가**를 “ trade-off “ 하는 것

→ 따라서 이 **trade-off**를 잘 조절해서 활용 가능 (*e.g. 가장 많이 사용되는 단어들의 pre-computation 결과만 가지는 등…*)

Sequence to sequence Learning: Conclusion

- 우리는 **character-level의 input만을 활용한** Neural Language Model을 소개하였음

  그리고, 이는 더 적은 parameter를 가지고 word/morpheme embedding을 사용한 모델들보다 좋은 성능을 보임

- 따라서, 우리의 연구는 *NLM을 위해 **과연 Word embedding이 필요할까?***라는 의문을 던짐

  → Character 조합으로 얻어진 단어 표현만으로도 **semantic, orthographic한 feature**를 뽑아낼 수 있음

- **CharCNN + highway network의 조합**은 앞으로도 연구가 더 수행되어야 할 하나의 방법인 것 같다 ! (?)

- Input layer에서의 Word embedding은 다양한 NLP task에 있어 굉장히 보편적인 작업이기 때문에,

  encoder/decoder를 사용하는 NMT에서도 우리가 제시한 model이 잘 작동하는지를 연구하는 것도 흥미로울 것 !

<DONE>