Comparative Study of Neural Language Models

Abhinash Khare

Soham Pal

Sruthi Gorantla

Today we will see...

- A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005 (Tseng et al.)
- Exploring the Limits of Language Modeling (Jozefowicz et al.)
- Character-Based Neural Machine Translation (Ling et al.)
- Preliminary experimental results.

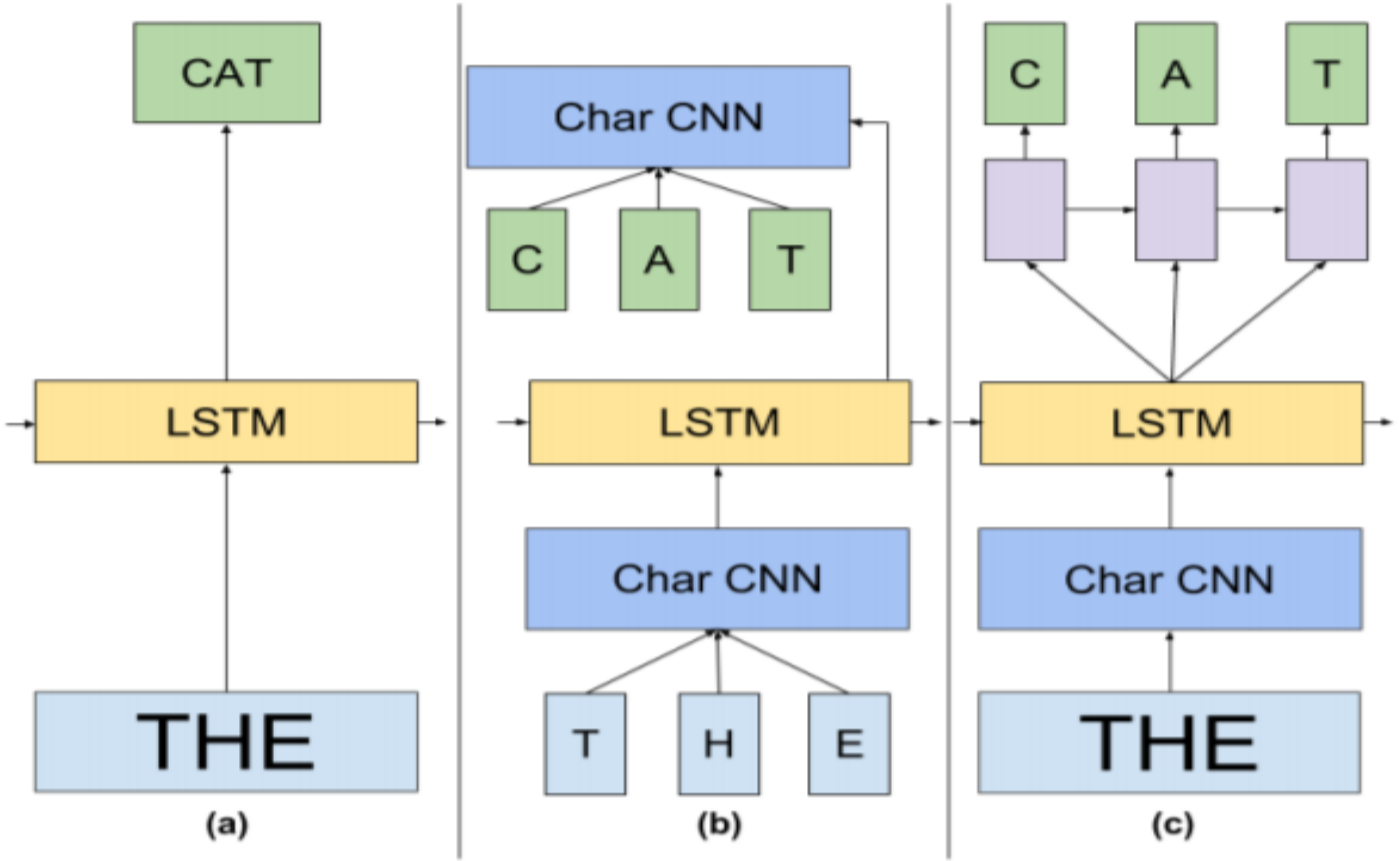
Exploring the Limits of Language Modeling

- Proposed by Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, Yonghui Wu from Google Brain in 2016.
- Performed an empirical study on Language Modeling using CNNs and RNN-LSTMs on One Billion Word Benchmark.
- Proposed models that improve the state-of-the-art perplexity significantly.

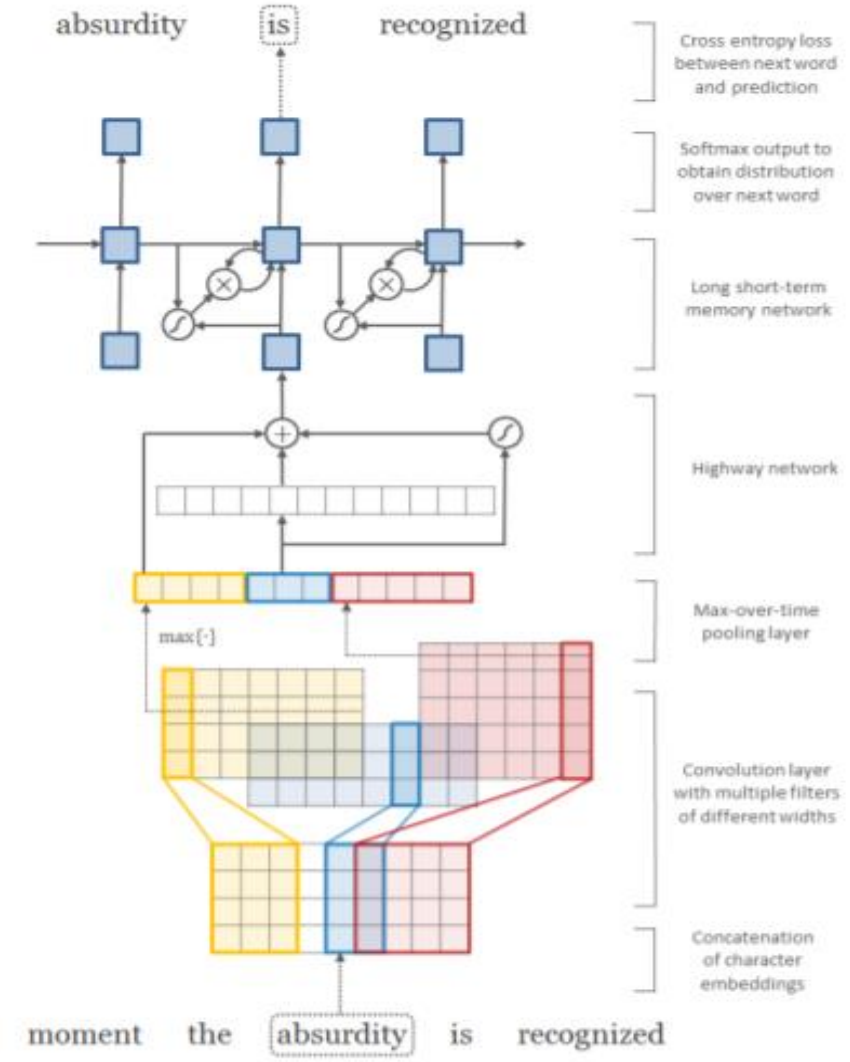
Introduction

- Believe that results on large datasets show much better about what actually matters.
- **Dataset:** One Billion Word Benchmark Dataset (800k word vocabulary).
- **Models:** They have considered three different models using LSTMs and Char CNNs.

Models (Jozefowicz et al.)



Char-CNN (Kim et al.)



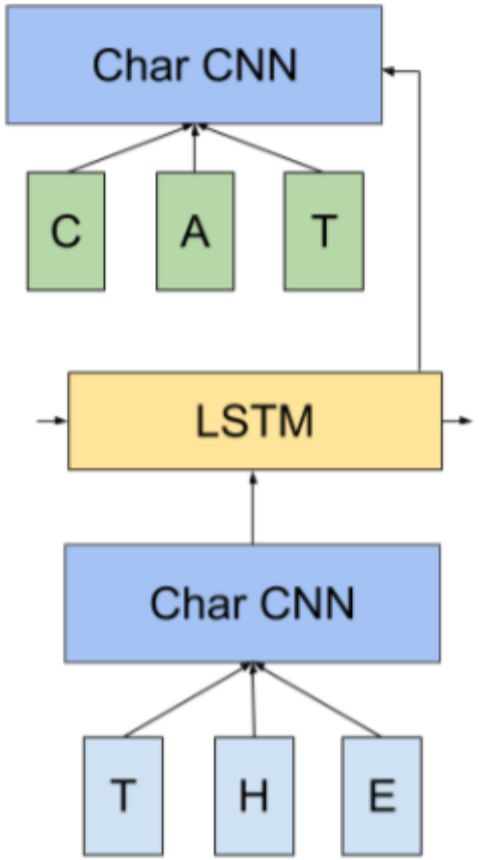
Challenges in Language Modeling

- Learning a very good model for human language which requires large models and thus, large amounts of data.
- Softmax over large vocabularies.

$$p(w) = \frac{\exp(z_w)}{\sum_{w' \in V} \exp(z_{w'})} \qquad \text{where} \qquad z_w = h^T e_w$$

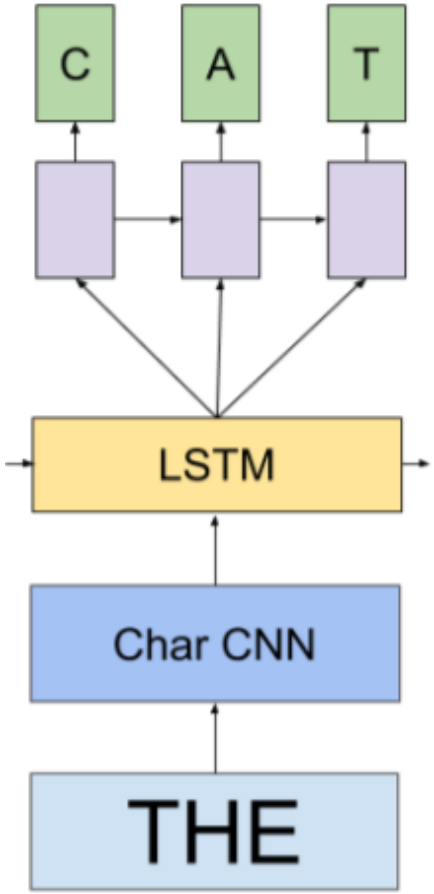
Improvements in Language Modeling (1)

- **CNN Softmax:**
  - Instead of computing a huge vector |V|*|h|, e_w is calculated by a CNN over the characters of w as e_w = CNN(chars_w).
  - Though the training time doesn't decrease much, the number of parameters is reduced as the CNN is shared for all the words and the computations can be parallelised.



Improvements in Language Modeling (2)

- **Char LSTM Predictions:**
  - Used to predict one character at a time such that the probabilities are calculated over much smaller vocabulary.
  - Combination of word-level and char-level LSTMs are used. However, they perform worse than the regular or CNN softmax, even though they scale independently of vocabulary size both during training as well as inference.



Results (Jozefowicz et al.)

Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN Inputs	**30.0**	**1.04**
BIG LSTM+CNN Inputs + CNN Softmax	39.8	**0.29**
BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction	35.8	**0.39**
BIG LSTM+CNN Inputs + Char LSTM predictions	47.9	**0.23**

Results for Ensemble of Models

Model	Test Perplexity
Large Ensemble (Chelba et al., 2013)	43.8
RNN+KN-5 (Williams et al., 2015)	42.4
RNN+KN-5 (Ji et al., 2015a)	42.0
RNN+SNM10-skip (Shazeer et al., 2015)	41.3
Large Ensemble (Shazeer et al., 2015)	41.0
Our 10 best LSTM models (equal weights)	26.3
Our 10 best LSTM models (optimal weights)	26.1
10 LSTMs + KN-5 (equal weights)	25.3
10 LSTMs + KN-5 (optimal weights)	25.1
10 LSTMs + SNM10-skip (Shazeer et al., 2015)	**23.7**

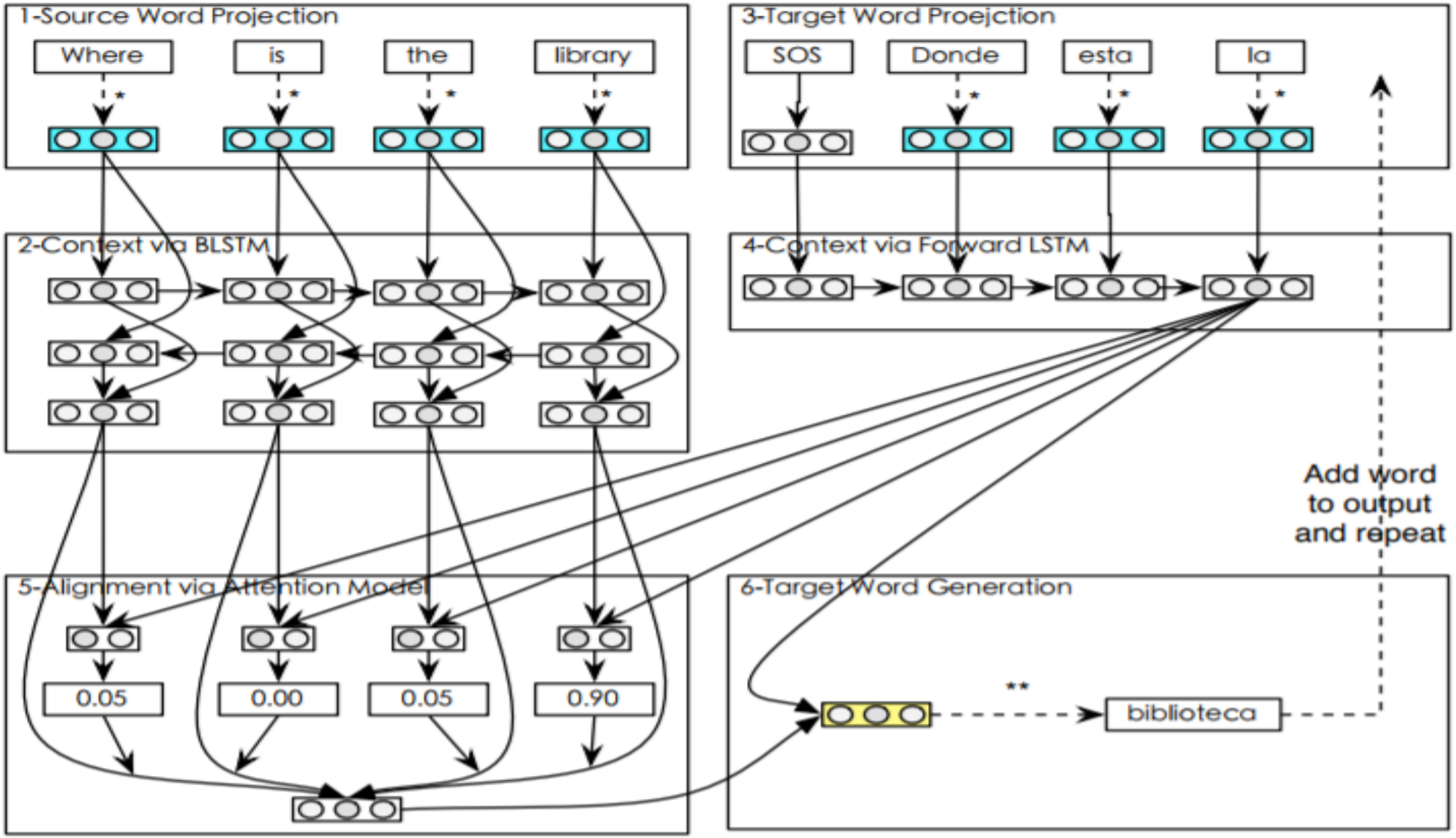
Key Observations

- Size matters.
- Regularization importance.
- Word Embeddings vs Character CNN.
- Smaller models with CNN Softmax.

Character-Based Neural Machine Translation

- Proposed by Wang Ling, Isabel Trancoso, Chris Dyer and Alan Black.
- The input and output sentences are sequences of characters rather than words.
- Proposed models are capable of generating and interpreting unseen word forms.
- Less data preprocessing of the source and target languages.

Joint Alignment and Translation Model



Character-Based Neural Machine Translation

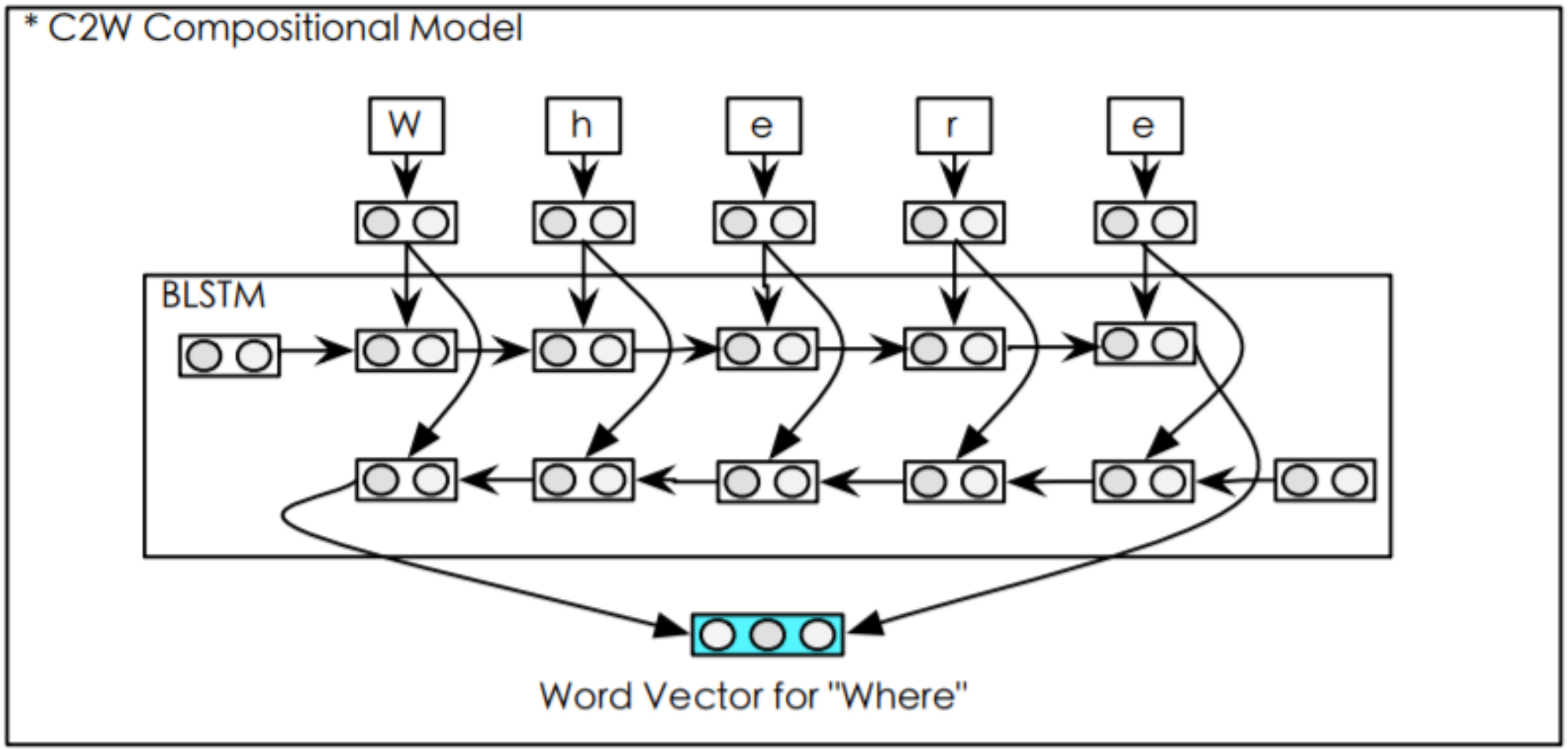
Alignment via Attention

- The model computes a score for each source word and a softmax over all the scores is performed.

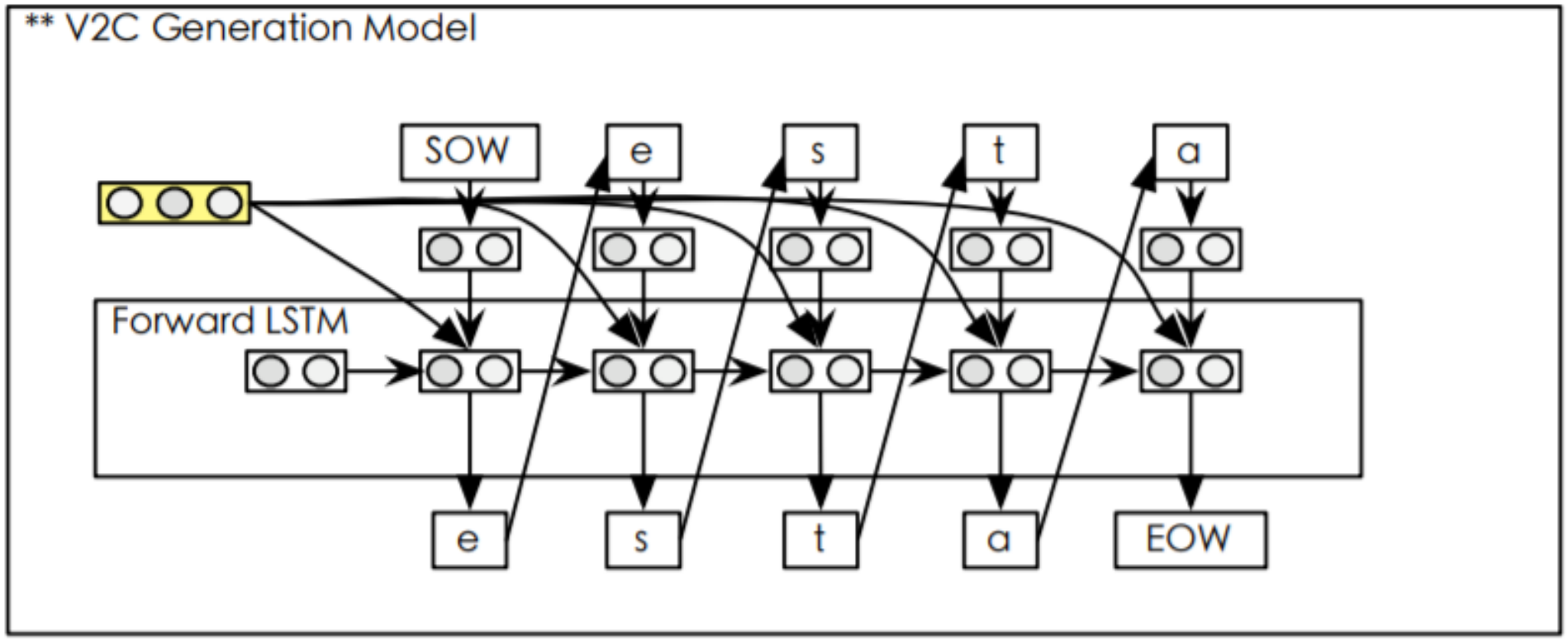
$$z_i = \mathbf{s} \tanh(\mathbf{W}_t \mathbf{l}^f_{p-1} + \mathbf{W}_s \mathbf{b}_i)$$

$$a_i = \frac{\exp(z_i)}{\sum_{j \in [0,n]} \exp(z_j)}$$

Character-Based Word Representation (C2W)



Character-Based Word Generation (V2C)



Layer-wise Training

- Firstly, characters are composed into word vectors using the C2W model.
- Then, the attention model searches for the next source word to translate.
- Finally, the generation of the target word is obtained using the V2C model

Results (1)

- C2W prefers to gather words that are orthographically similar.

C2W Model			Word Lookup Table		
answer	*well-founded*	*responder*	*answer*	*well-founded*	*responder*
response	well-balanced	respondeu	reply	described	reagir
answers	much-needed	responderam	response	impressed	aderir
reply	self-employed	responda	answering	bizarre	responda
answered	uncontrolled	responde	join	santer	aceder
replies	inherited	respondem	question	unclear	agradecer

Ref: Character-Based Neural Machine Translation (Ling et al.)

Results (2)

- V2C model generates unseen words.

original	. . . that does not mean that we want to bring an end to **subsidisation** .
character translation	. . . isso não significa a questão de que se trata de um fim à **subsidade** .
word translation	... Isso não significa que isso , para conseguir reduzir os **autores** .
original	the budget for the **reconstruction** of ...
character translation	o orçamento para as **reconstruções** ...
word translation	o orçamento inerente à **reconstrução** ...

Ref: Character-Based Neural Machine Translation (Ling et al.)

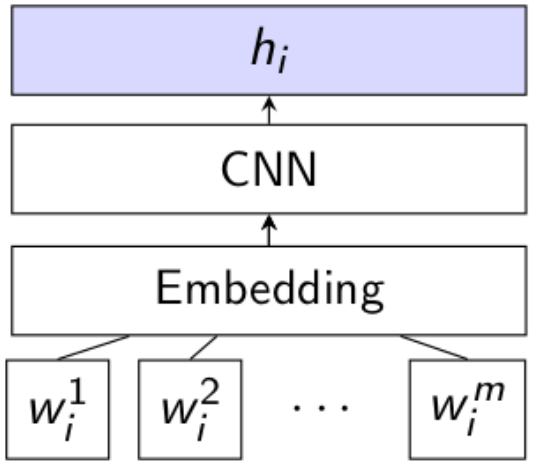
Conclusion

- Using characters as atomic units helps in learning orthographically sentient word representations.
- Unseen word forms can be generated and thus, showing that character-based neural translation models perform better than word-based models.

Review

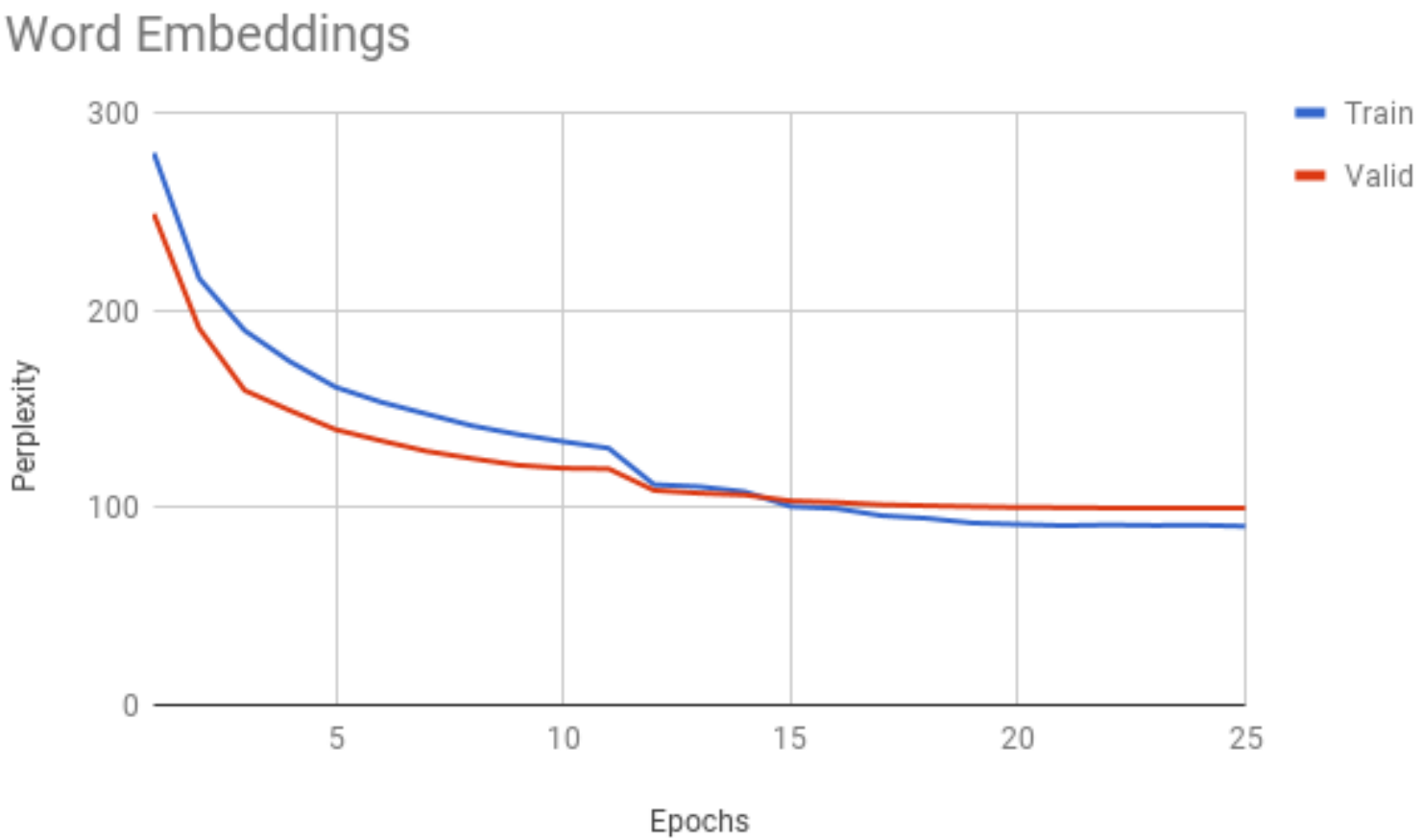
Our 4 proposed models:

- Word embeddings
- Concatenation of character embeddings
- CNN on character embeddings
- RNN on character embeddings

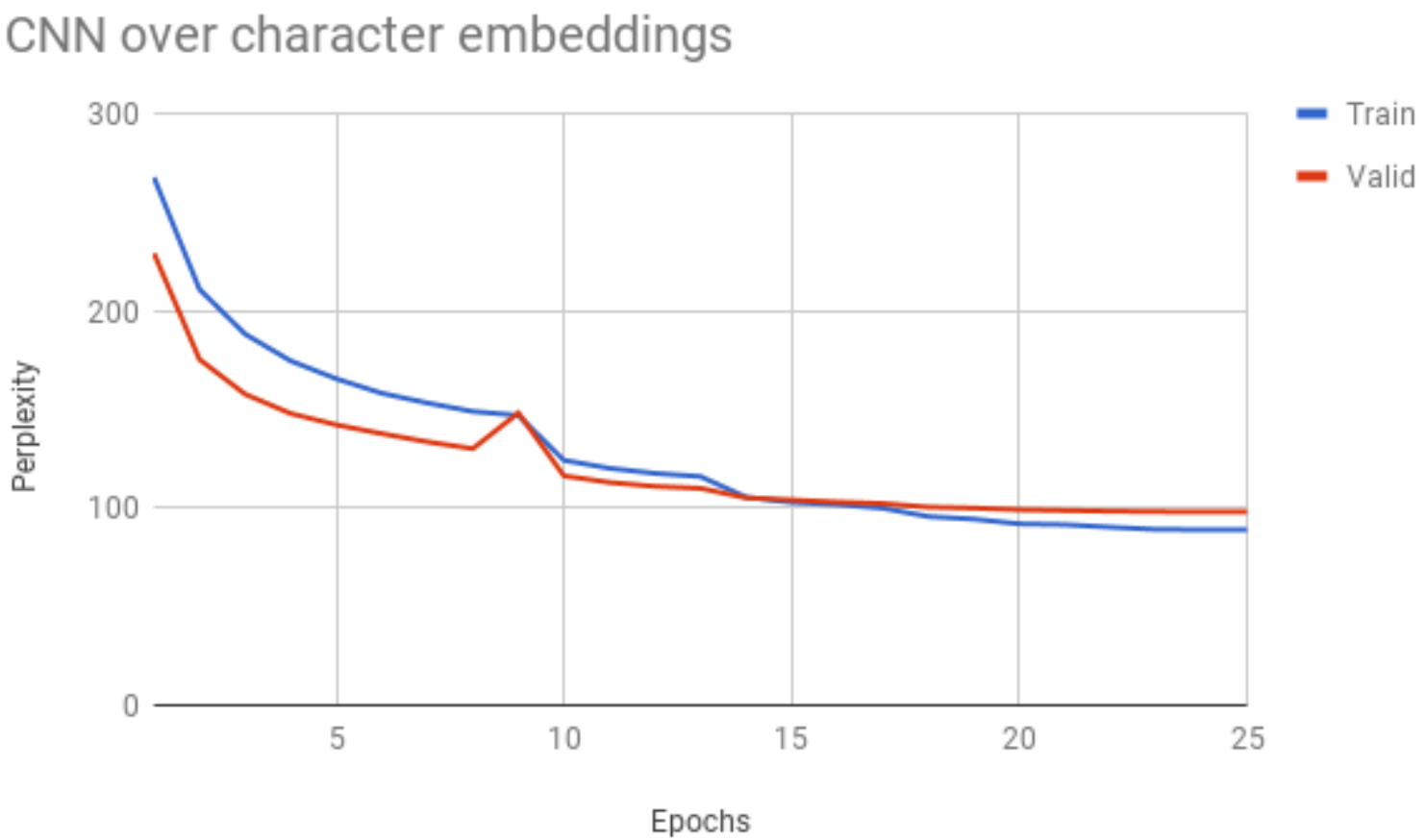


CNN on characters

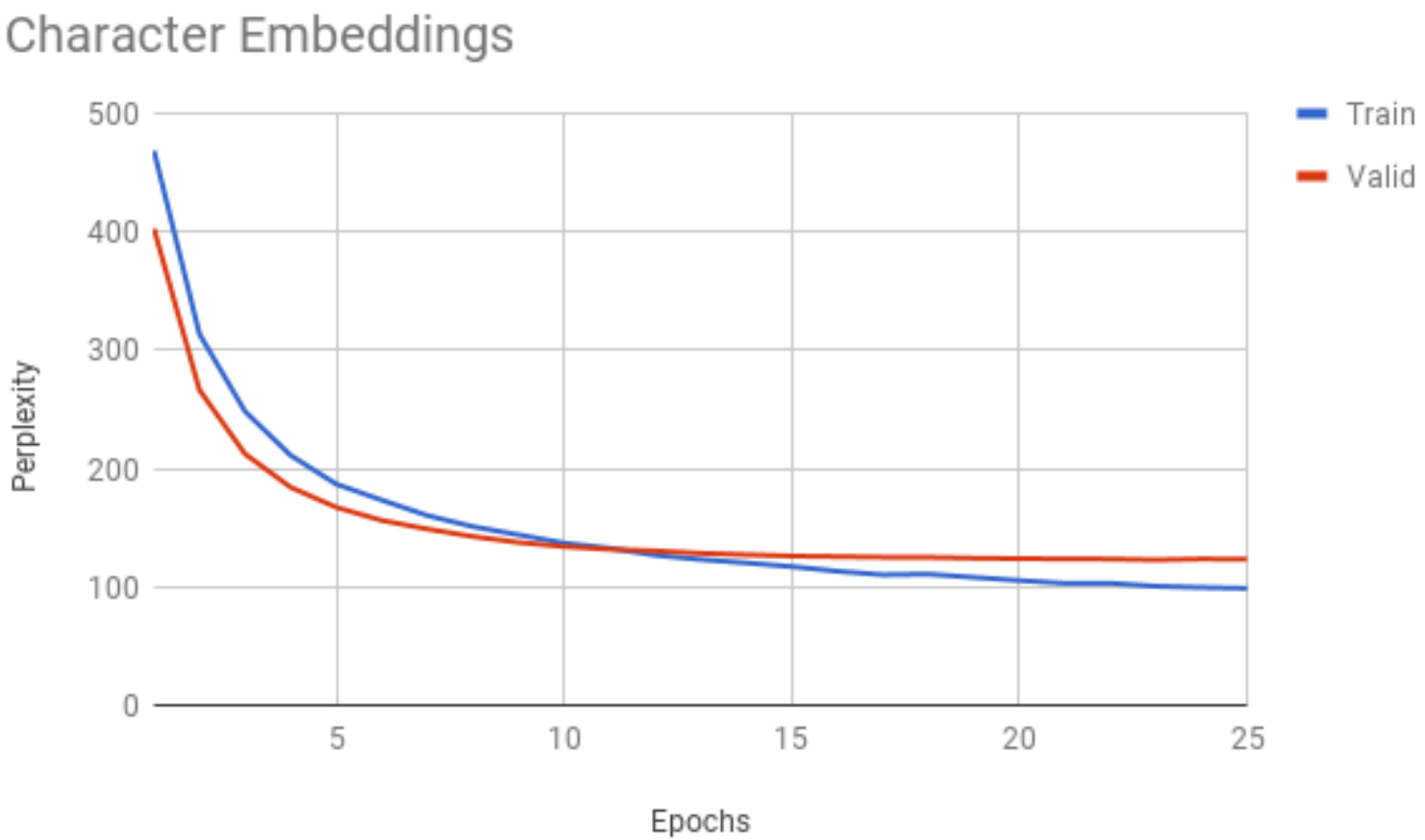
Word embeddings



CNN over character embeddings



Concatenated character embeddings



Preliminary Results

- Results reported on the **English Penn Treebank** (PTB), and compared with *Character-Aware Neural Language Models* (Yoon Kim).

Model	Reported	Reproduced	Our Implementation
Word embeddings	97.6	95.814389158859	94.82428529
Concatenation of character embeddings	N/A	N/A	114.72031261
CNN over character embeddings	92.3	92.6425160752	94.28213722