# Correcting Language Errors using Machine Translation Techniques


Shamil Chollampatt

shamil@u.nus.edu

Natural Language Processing Group

National University of Singapore

# What are language errors?

- A “language error” is a deviation from rules of a language
- Due to lack of knowledge.
- Made by learners of the language.
- Language errors in writing include spelling, grammatical, word choice, and stylistic errors

# How can NLP help?

- Building automatic grammar correction tools and spell checkers.
- Rule-based systems (e.g. Microsoft Word), and advanced software that correct different kinds of errors (e.g. Grammarly, Ginger).
- Useful tool for non-native writers.
- Evidence that corrective feedback helps language learning (Leacock et al., *Automated Grammatical Error Detection for Language Learners* 2ed, 2014)

# Grammatical Error Correction or “GEC”

- Automatic correction of various kinds of errors in written text.

Example (input):

*The problems ~~bring some effect on~~ affect engineering design ~~from~~ **in** two ~~aspect~~ **aspects**, independent innovation and engineering application.*

– from the NUS Corpus of Learner English (NUCLE)

- Most popular approach is the machine translation approach.

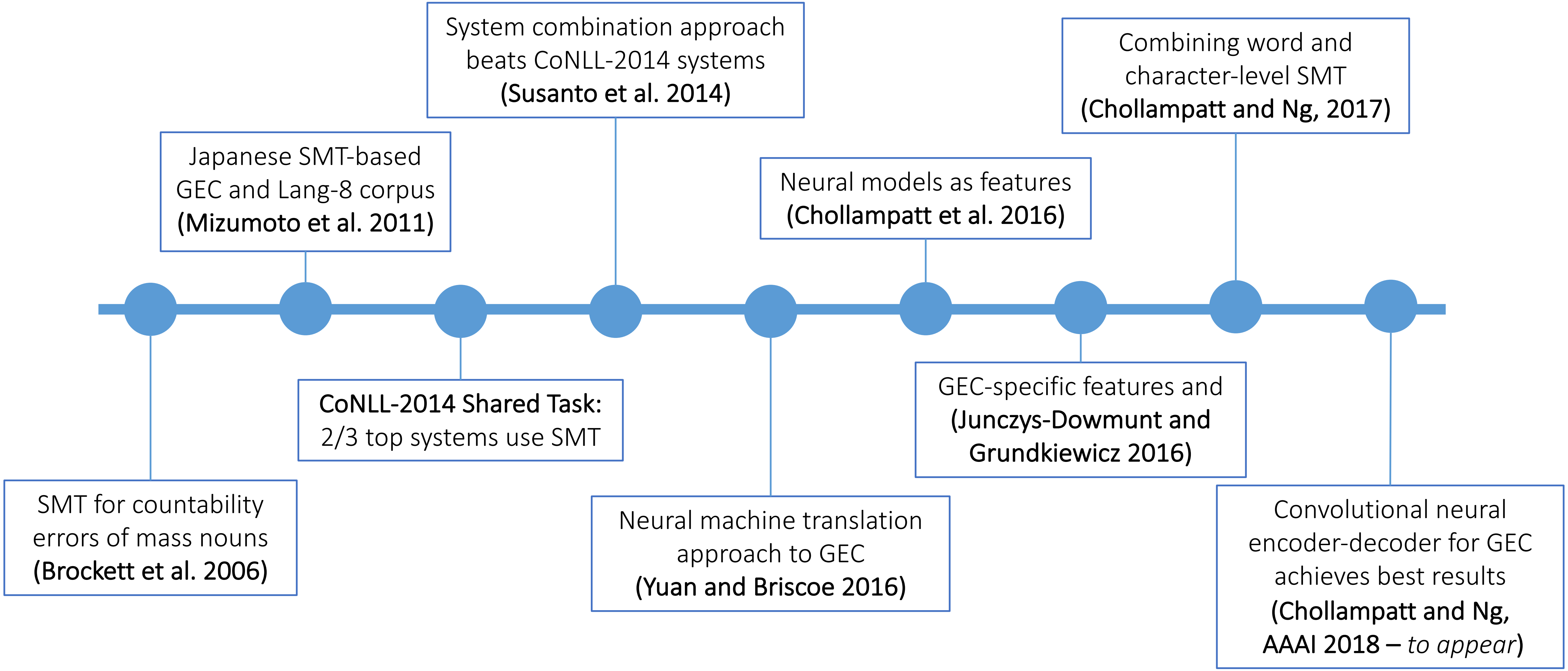
# The Translation Approach

- Treats GEC as ***translation*** task from
  "bad" English → "good" English

  Advantages:
  ✓Able to learn text transformations from parallel data.
  ✓Simple, and does not need language-dependant tools.
  ✓Can correct interacting errors and complex error types.

- Typically, statistical machine translation (SMT) or neural machine translation (NMT) frameworks.

# History


SMT for countability errors of mass nouns **(Brockett et al. 2006)**
Japanese SMT-based GEC and Lang-8 corpus **(Mizumoto et al. 2011)**
**CoNLL-2014 Shared Task:** 2/3 top systems use SMT
System combination approach beats CoNLL-2014 systems **(Susanto et al. 2014)**
Neural machine translation approach to GEC **(Yuan and Briscoe 2016)**
Neural models as features **(Chollampatt et al. 2016)**
GEC-specific features and **(Junczys-Dowmunt and Grundkiewicz 2016)**
Combining word and character-level SMT **(Chollampatt and Ng, 2017)**
Convolutional neural encoder-decoder for GEC achieves best results **(Chollampatt and Ng, AAAI 2018 – *to appear*)**

# Data

## For training:

- *Parallel Corpora*
  - Annotated Learner Dataset: NUCLE
  - Crawled from Lang-8
- *English corpora*:
  Wikipedia, CommonCrawl

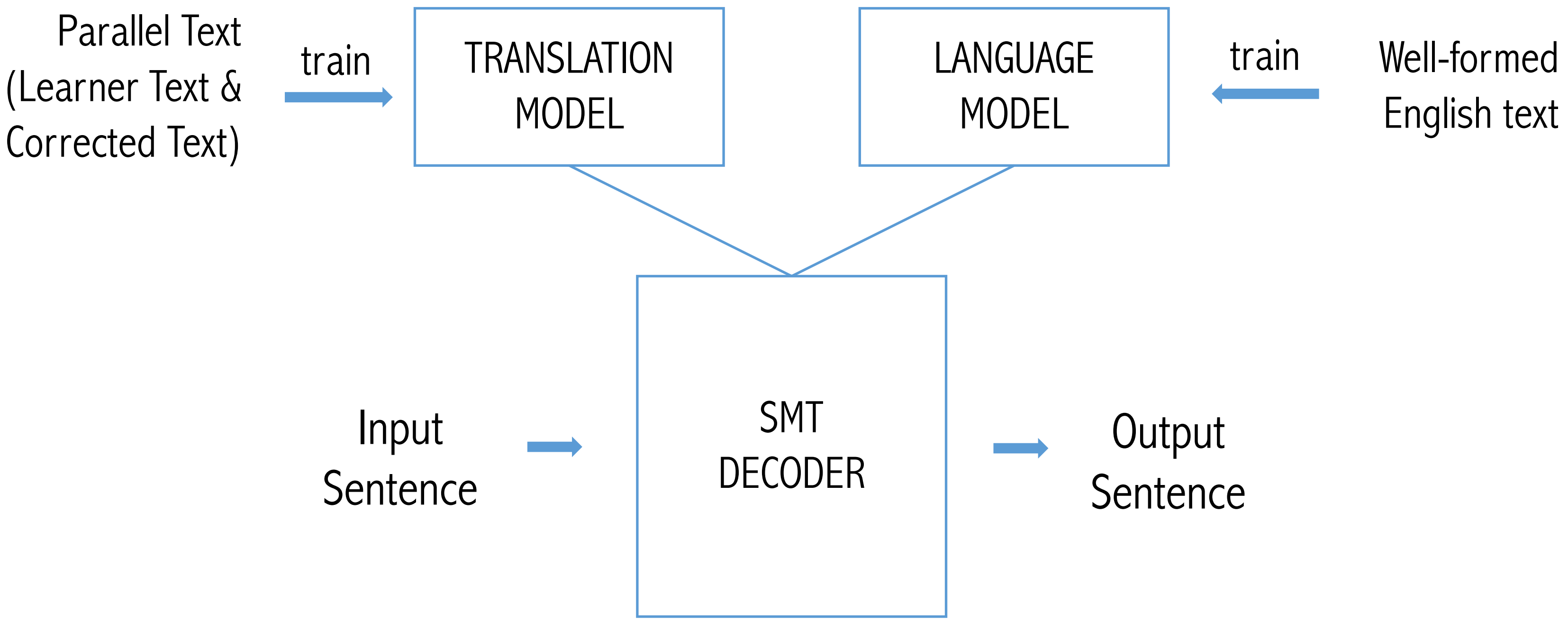

## For testing:

CoNLL-2014 shared task test set (1312 sentences)

*Metric*: $F_{0.5}$ using MaxMatch scorer

# Word and Character-level SMT for GEC

# Statistical Machine Translation Approach



# Statistical Machine Translation Approach

- Using a log-linear framework:

$$T^* = \underset{T}{\operatorname{argmax}}\, P(T|S) = \underset{T}{\operatorname{argmax}} \sum_{i=1}^{N} \lambda_i \left(f_i(S,T)\right)$$

$T^*$ : best output sentence
$S$ : source sentence
$T$ : candidate output sentence
$N$ : number of features
$\lambda_i$ : $i^{th}$ feature weight
$f_i$ : $i^{th}$ feature function

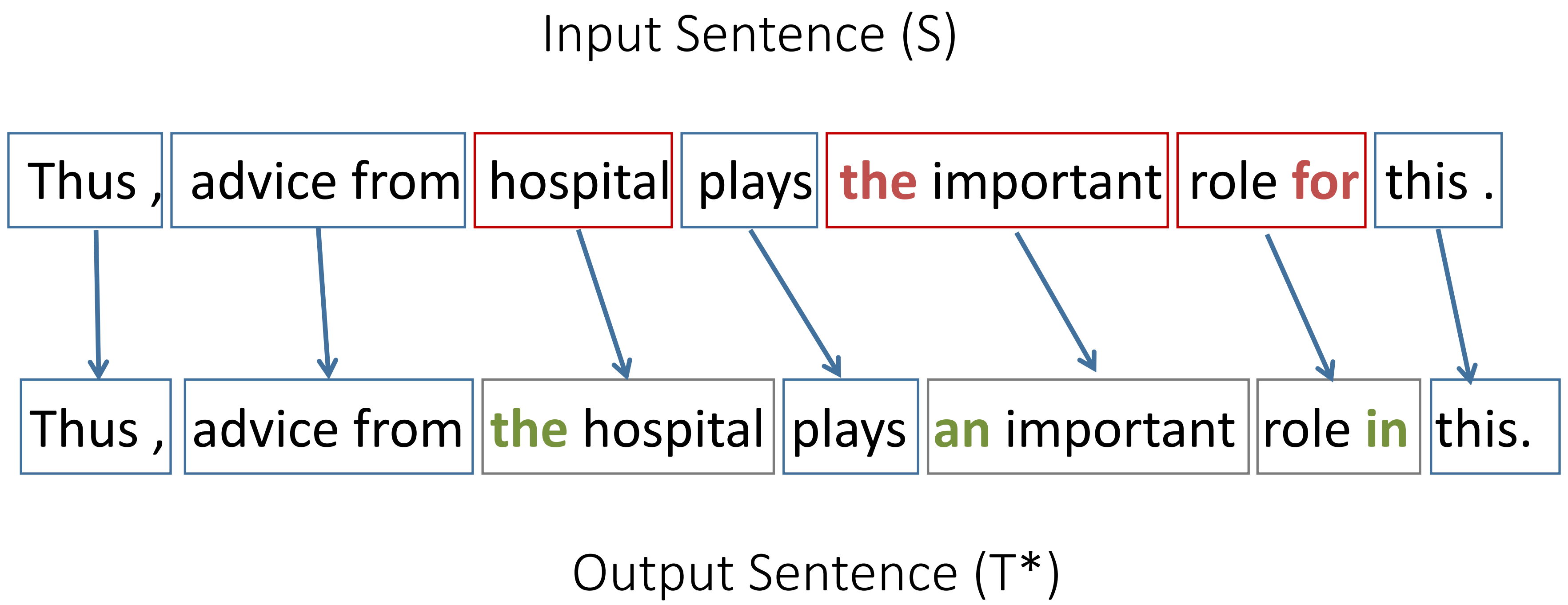
- Feature weights $\lambda_i$ are tuned using MERT optimizing $F_{0.5}$ metric on development set.

# Phrase-based SMT

Input Sentence (S)

Thus , advice from hospital plays **the** important role **for** this .

# Phrase-based SMT



# Useful GEC-specific Features

- Introduced by Junczys-Dowmunt and Grundkiewicz (CoNLL-2014 Shared Task, EMNLP 2016)
  - Word Class Language Model
  - Operation Sequence Model
  - Edit Operations
  - Sparse Edit Operation Features
  - A Web-scale LM

# Neural Network Joint Model

- Joint Model (JM) vs Language Model (LM)

SRC: The $\boxed{\text{cat sit in}}$ a mat .

HYP: The $\boxed{\text{cats sat}}$ on the mat .

3+2 gram JM: $P(\textbf{sat}|\text{cat, sit, in, cats})$

Bigram LM: $P(\textbf{sat}|\text{cats})$

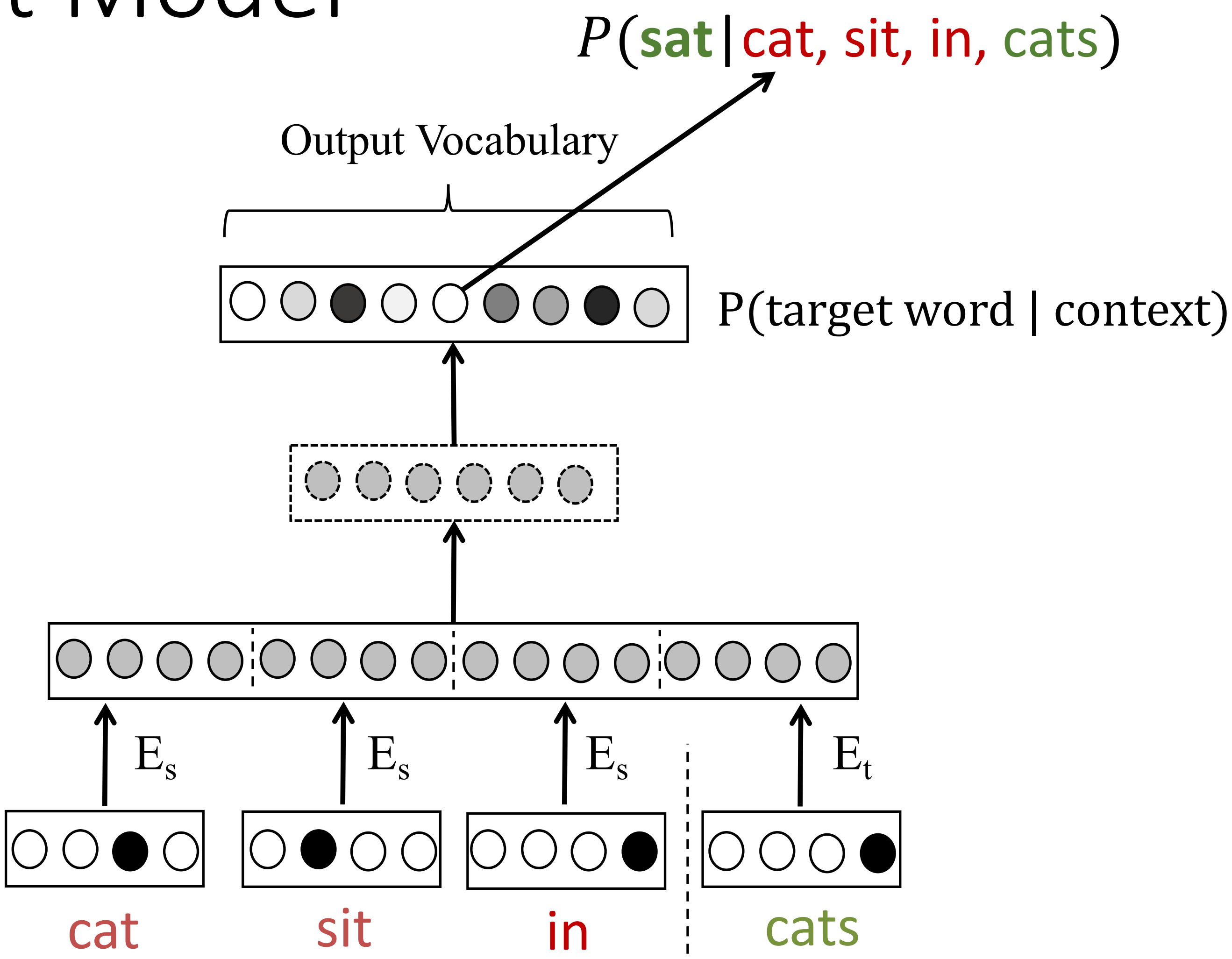
- Feature Function:

$$f(T,S) = P(T|S) \approx \prod_{i=1}^{|T|} P(t_i|s_{a-1}, s_a, s_{a+1}, t_{i-1})$$

# Neural Network Joint Model

- Uses a feed-forward neural network (Devlin et al., 2014)

- 5+5 gram NNJM for GEC in Chollampatt et al. (IJCAI 2016 and BEA Workshop 2017)



# NNJM Adaptation

Training: using log likelihood with self normalization.

$$L = \frac{1}{N}\sum_{i=1}^{N}\left[\log P(y = t_i|h_i) - \alpha \log^2(Z(h_i))\right]$$

Adaptation: adding KL-divergence regularization term to loss function:

$$K = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{V_o} P^{GD}(y = t_j|h_i) \log P(y = t_j|h_i)$$

<u>Adaptation Data:</u>

✓Higher quality error annotations

✓Higher error/sentence ratio

# SMT for Spelling Correction

- Added as a post processing step to the word-level SMT.

- Character-level SMT gets the unknown words from the SMT system and generates candidates (may be non-words)

utlises → utilises
utlises → utilizes
utlises → utilise
utlises → utilishes

- Rescoring with language model to filter away non-word candidates and pick best correction based on context.

# Setup

Development Data:

- 5,458 sentences from NUCLE with at least 1 error/sentence.

Parallel Training Data for Word-level SMT:

- Lang-8 , NUCLE (2.21M sentences, 26.77M source words)

Data for Character-level SMT:

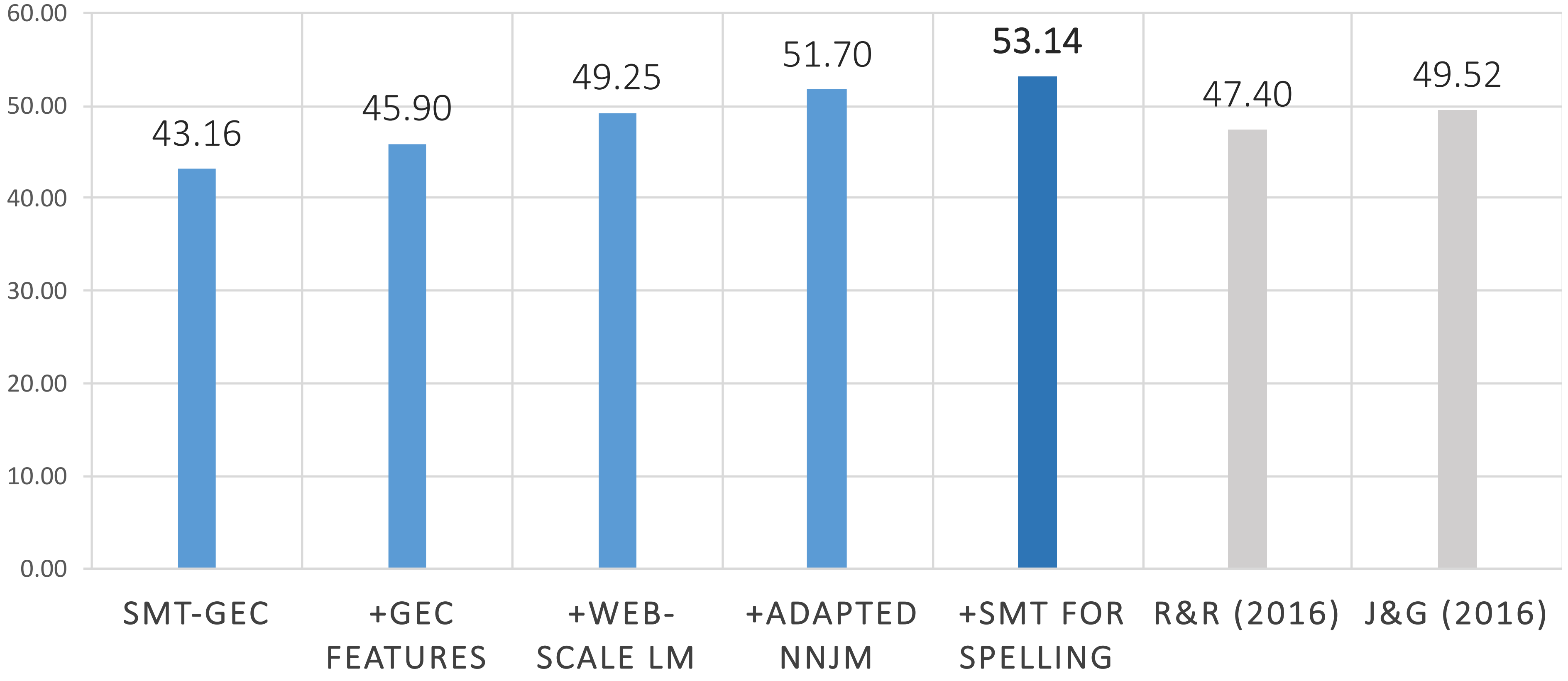
- Unique words in the corrected side of NUCLE and the corpora of misspellings (http://www.dcs.bbk.ac.uk/~ROGER/corpora.html)

LM Training Data:

- Wikipedia (1.78B tokens), Common Crawl LM (94B tokens)

# Results

R&R (2016) :ROZOVSKAYA AND ROTH (ACL 2016)
J&G (2016) :JUNCZYS DOWMUNT AND GRUNDKIEWICZ (EMNLP 2016)

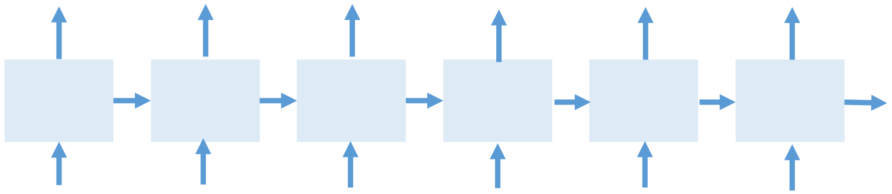


# Multilayer Convolutional Encoder and Decoder Neural Network for GEC

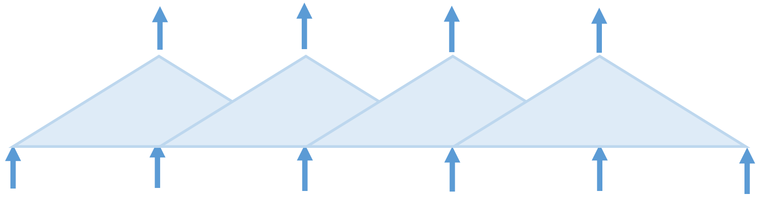
# Encoder-Decoder Approach

# Encoder-Decoder Approach

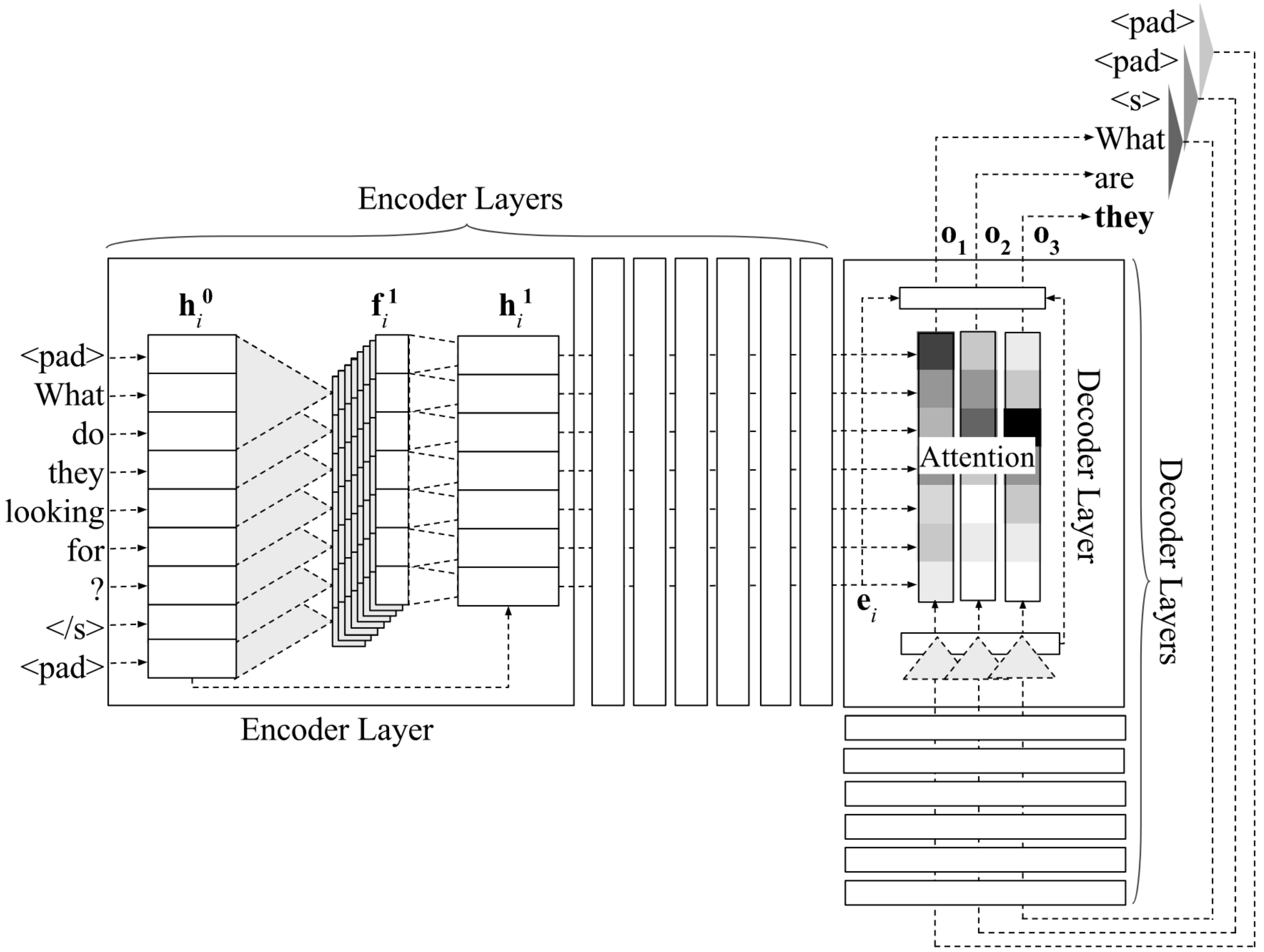
**Prior work in GEC**: Recurrent Neural Network (RNN)-based approaches (Bahdanau et al. 2015)



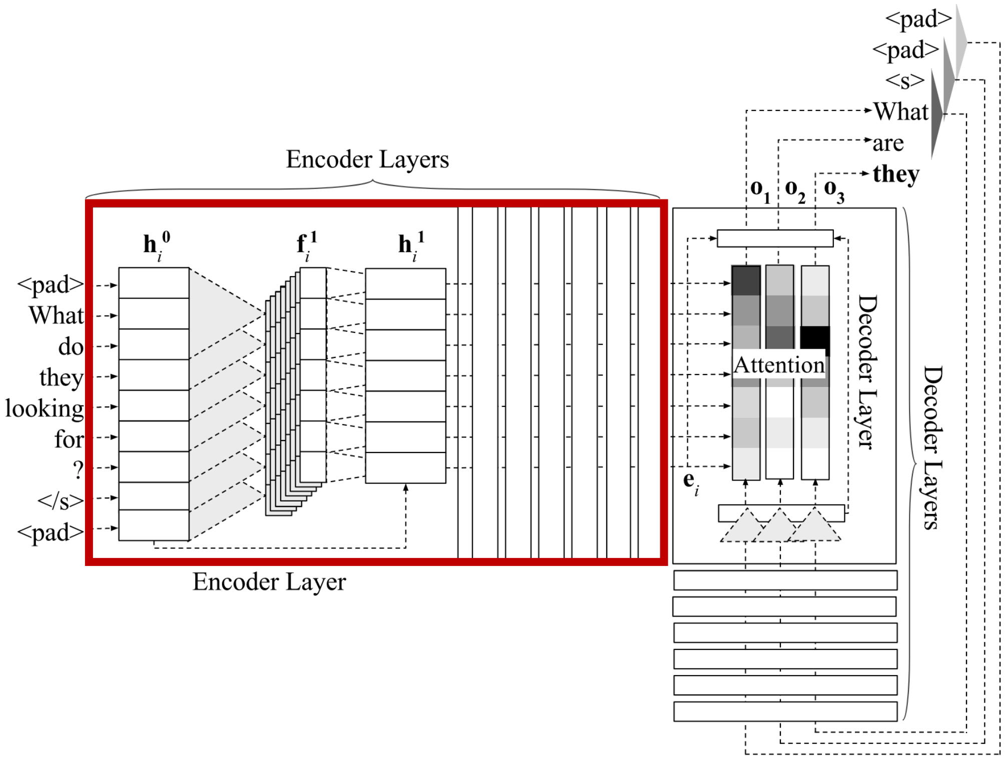
We use a fully Convolutional Neural Network (CNNs)-based approach (Gehring et al. 2017)...



# A Multilayer Convolutional Encoder-Decoder



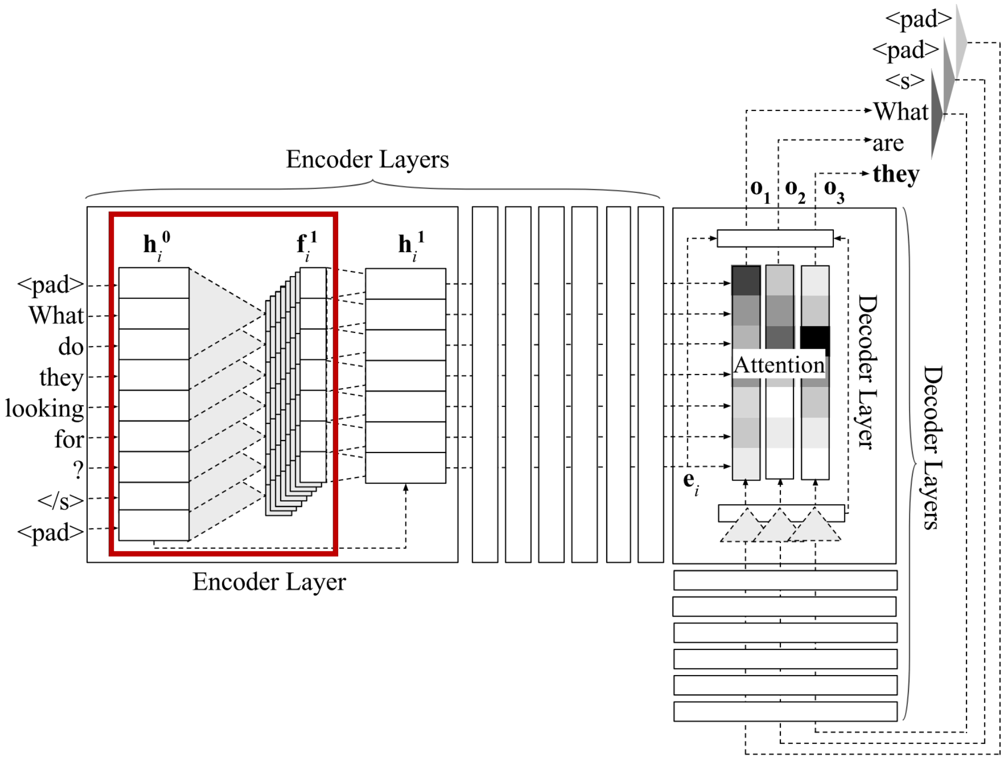
# A Multilayer Convolutional Encoder-Decoder




**Encoder**

Consists of seven layers.

# A Multilayer Convolutional Encoder-Decoder



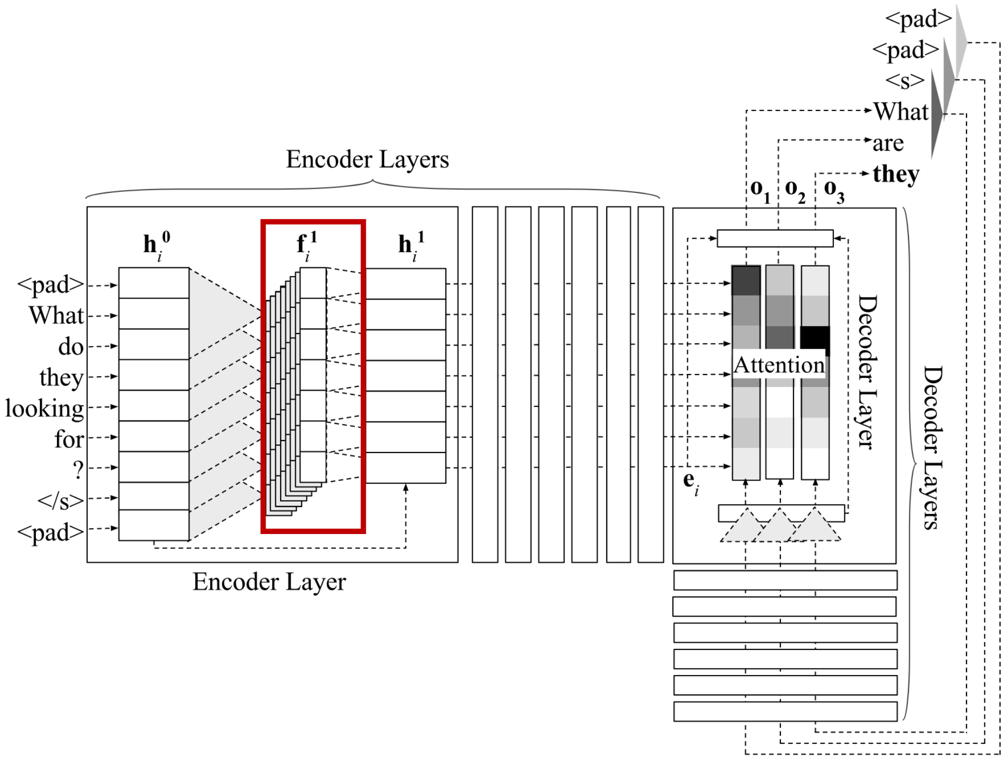

**Encoder**

Consists of seven layers.

Convolution Operation:

$$\mathbf{f}_i^l = \mathrm{Conv}\left(\mathbf{h}_{i-1}^{l-1}, \mathbf{h}_i^{l-1}, \mathbf{h}_{i+1}^{l-1}\right)$$

# A Multilayer Convolutional Encoder-Decoder




**Encoder**

Consists of seven layers.

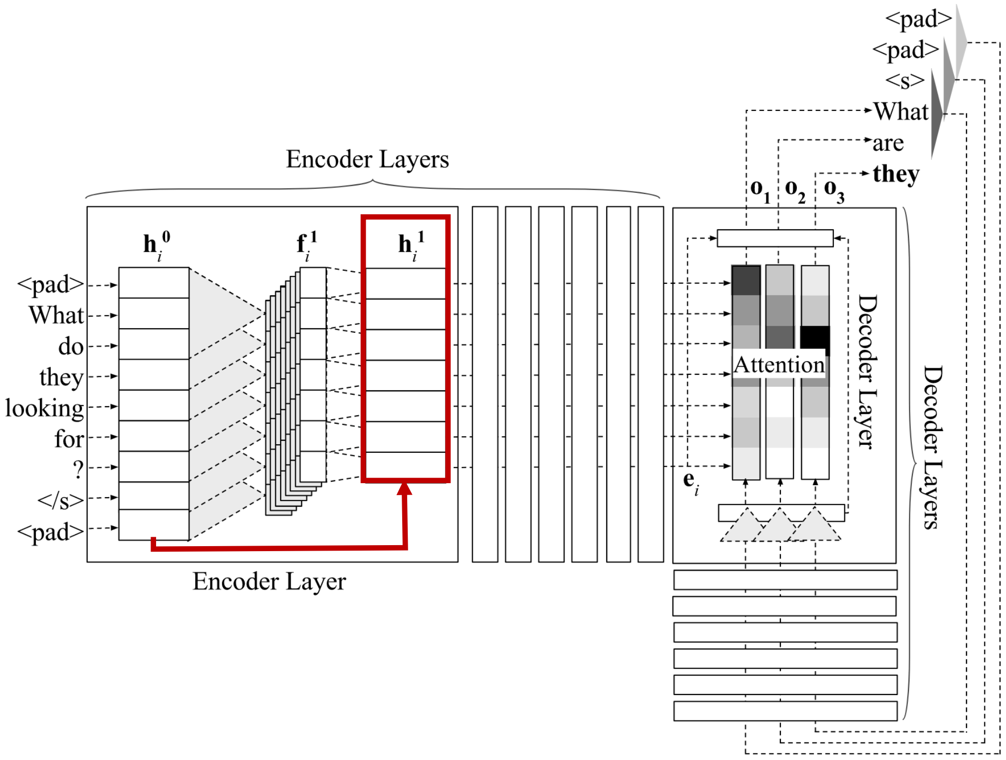
Convolution Operation:

$$\mathbf{f}_i^l = \mathrm{Conv}\left(\mathbf{h}_{i-1}^{l-1}, \mathbf{h}_i^{l-1}, \mathbf{h}_{i+1}^{l-1}\right)$$

Gated Linear Units (GLUs):

$$\mathrm{GLU}\left(\mathbf{f}_i^l\right) = \mathbf{f}_{i,1:h}^l + \sigma\left(\mathbf{f}_{i,h+1:2h}^l\right)$$

# A Multilayer Convolutional Encoder-Decoder




**Encoder**

Consists of seven layers.

Convolution Operation:

$$\mathbf{f}_i^l = \mathrm{Conv}\left(\mathbf{h}_{i-1}^{l-1}, \mathbf{h}_i^{l-1}, \mathbf{h}_{i+1}^{l-1}\right)$$

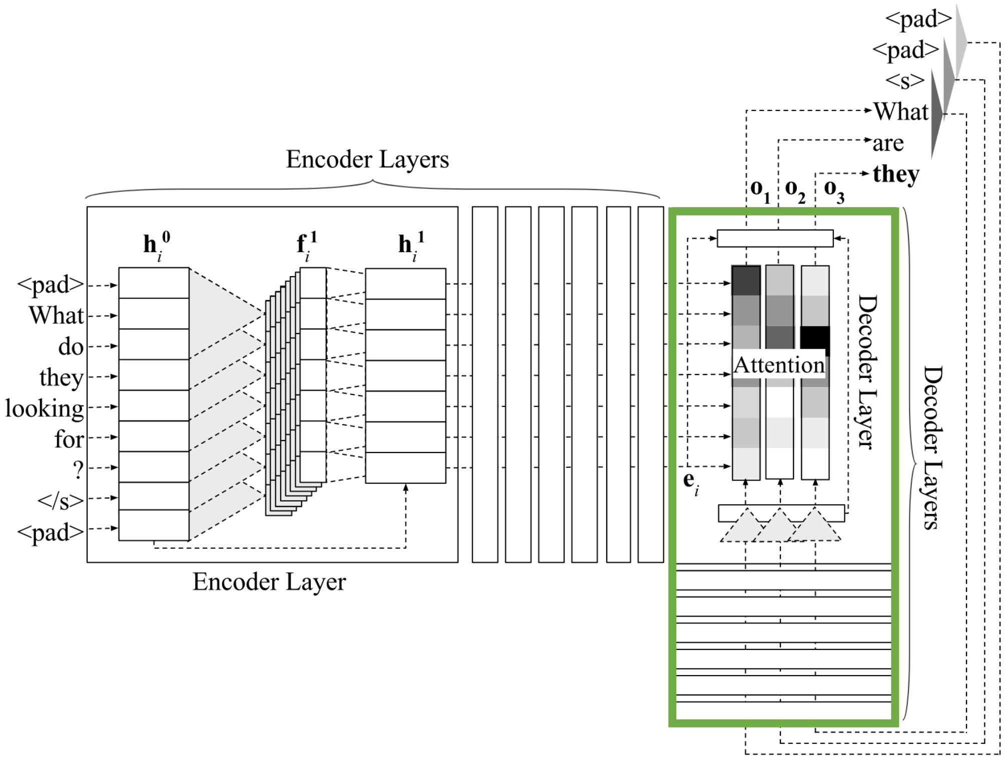
Gated Linear Units (GLUs):

$$\mathrm{GLU}\left(\mathbf{f}_i^l\right) = \mathbf{f}_{i,1:h}^l + \sigma\left(\mathbf{f}_{i,h+1:2h}^l\right)$$

Residual Connections:

$$\mathbf{h}_i^l = \mathrm{GLU}\left(\mathbf{f}_i^l\right) + \mathbf{h}_i^{l-1}$$

# A Multilayer Convolutional Encoder-Decoder

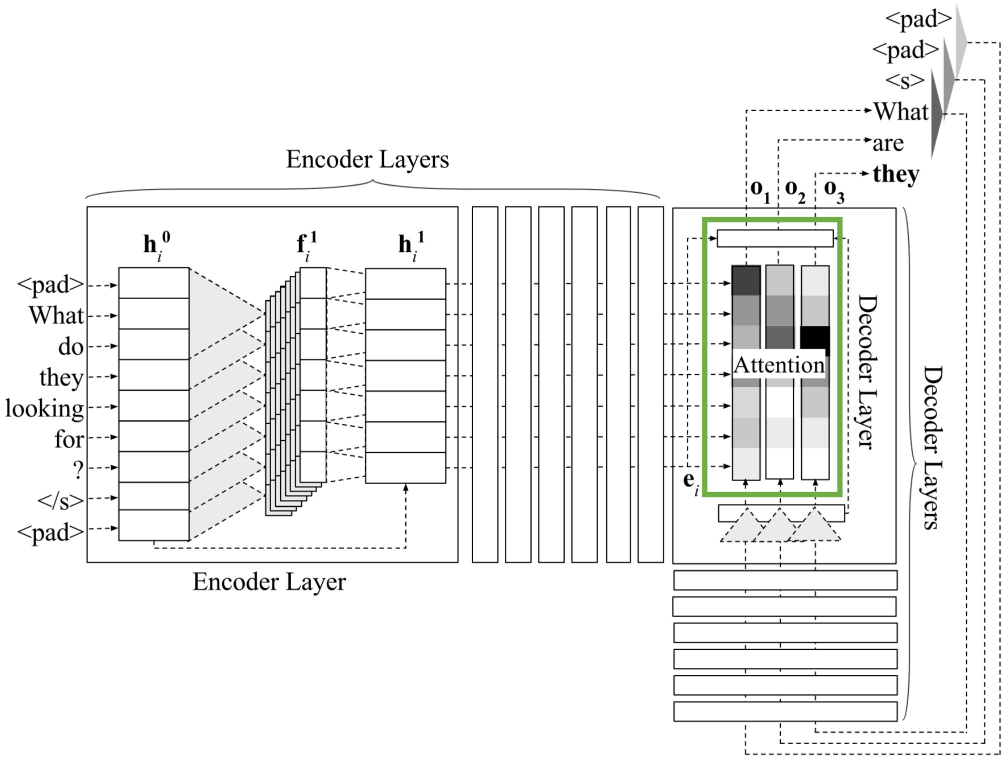



**Decoder**

Consists of seven layers.

Consists of convolutions and non-linearities

# A Multilayer Convolutional Encoder-Decoder




**Decoder**

Consists of seven layers.

Consists of convolutions and non-linearities

+ Attention:

$$\alpha_{n,i}^{l} = \frac{\exp(\mathbf{e}_i^{\mathrm{T}}\mathbf{z}_n^{l})}{\sum_{k=1}^{m}\exp(\mathbf{e}_k^{\mathrm{T}}\mathbf{z}_n^{l})}$$

$$\mathbf{x}_n^{l} = \sum_{i=1}^{m}\alpha_{n,i}^{l}\,(\mathbf{e}_i + \mathbf{s}_i)$$

# Pre-training Word Embeddings

- Word embeddings are pre-trained and initialized.
- Trained using *fastText* (Bojanowski et al., 2017) on Wikipedia.
- Uses underlying character n-gram sequences of words

Advantages

✓Reliable embeddings can be constructed for rarer words.

✓Morphology of words is considered.

# Ensembling and Re-scoring

- Ensembling multiple models, i.e. the log probabilities for multiple models are averaged during prediction of each output word.

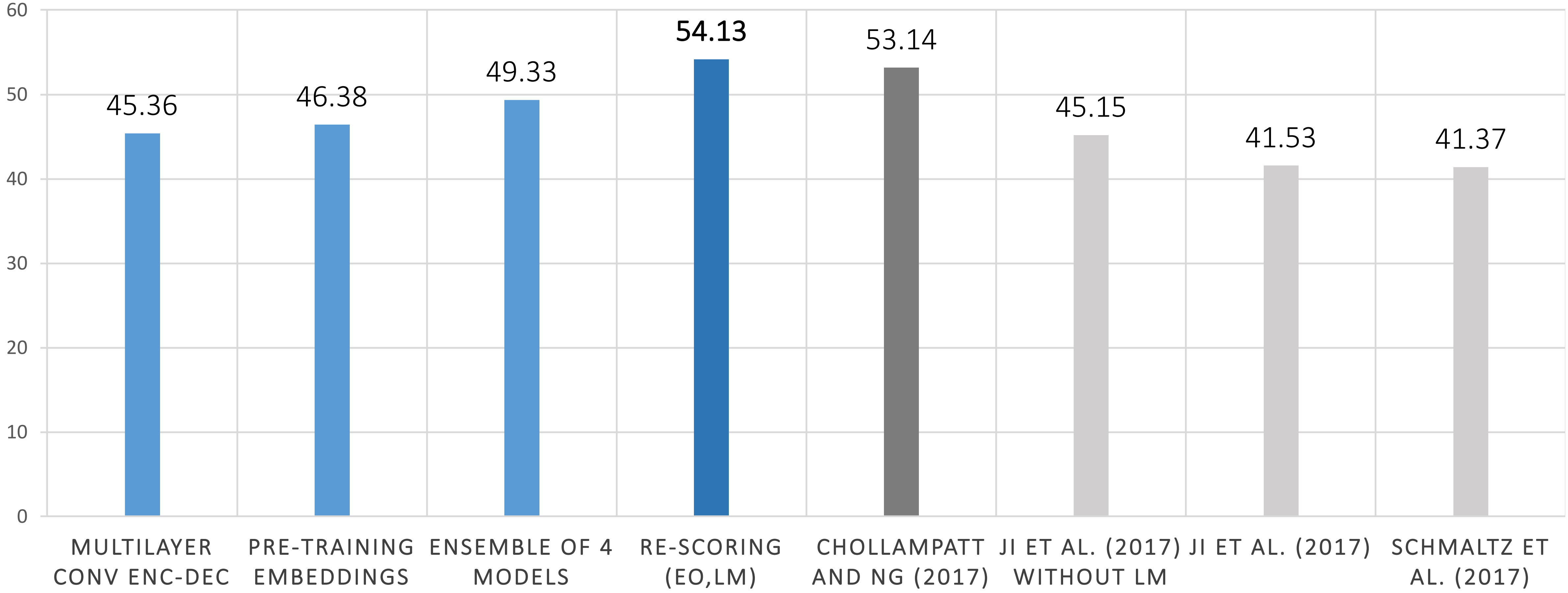
- The final beam candidates are re-scored using features:
  - **Edit Operation (EO):** #insertions, #deletions, #substitutions
  - **Language Model (LM):** web-scale LM score, #words

- Feature weights tuning done similar to SMT: MERT optimizing $F_{0.5}$ on the development data.

# Model and Training Details

- Data: As in Chollampatt and Ng (BEA 2017) except for using only annotated sentence pairs during training.
- Vocabulary: 30K most frequent words on source and target side
- Number of dimensions of embeddings: 500
- Number of dimensions of encoder/decoder output vectors: 1024

# Results


60
50
40
30
20
10
0
45.36
46.38
49.33
54.13
53.14
45.15
41.53
41.37
MULTILAYER CONV ENC-DEC
PRE-TRAINING EMBEDDINGS
ENSEMBLE OF 4 MODELS
RE-SCORING (EO,LM)
CHOLLAMPATT AND NG (2017)
JI ET AL. (2017) WITHOUT LM
JI ET AL. (2017)
SCHMALTZ ET AL. (2017)

# Challenges and Future Work

- Lack of good quality parallel data.
- Going beyond sentence-level.
- Adaptation to diverse learners.

# Thank You

**Email**: shamil@u.nus.edu
**Website**: shamilcm.github.io