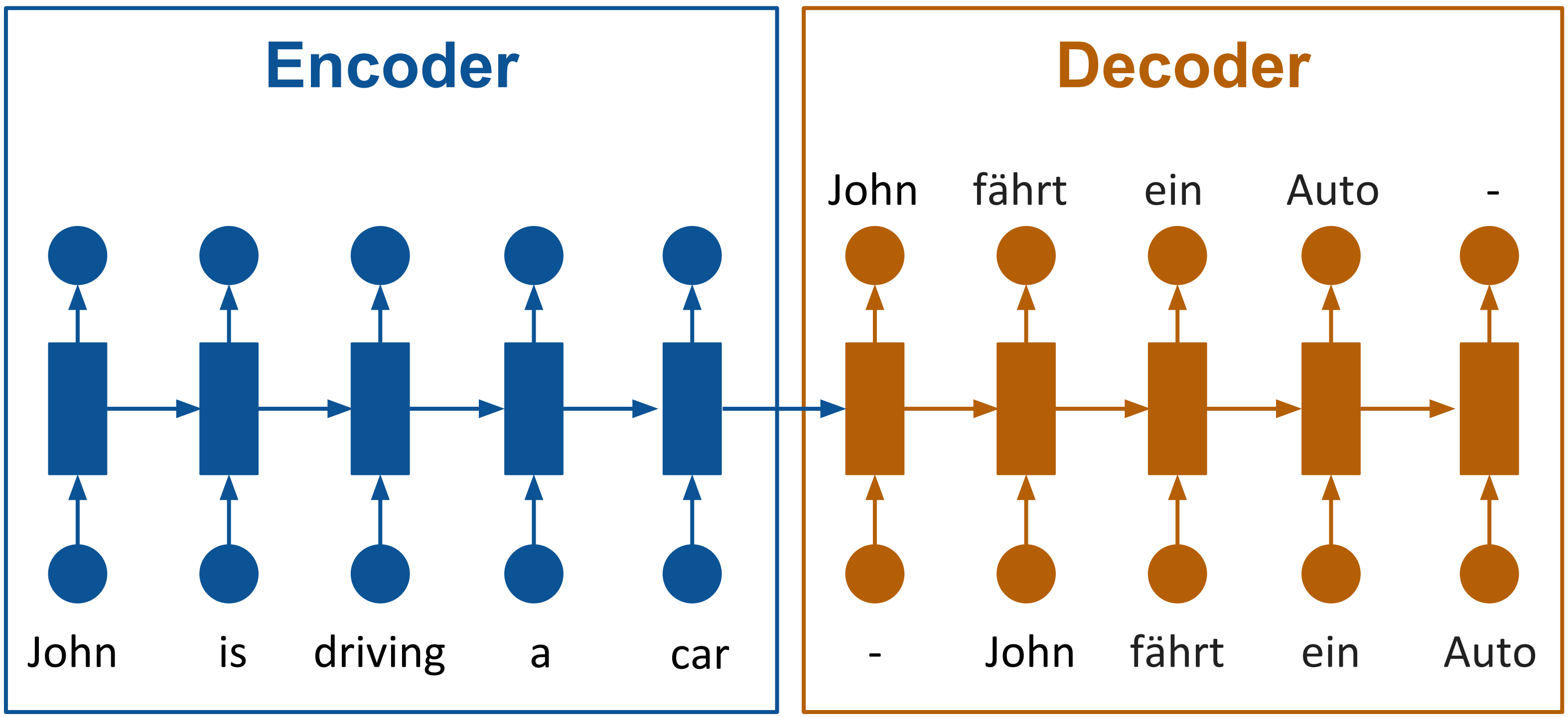
Neural Machine Translation: Practical Considerations

Lecture # 8

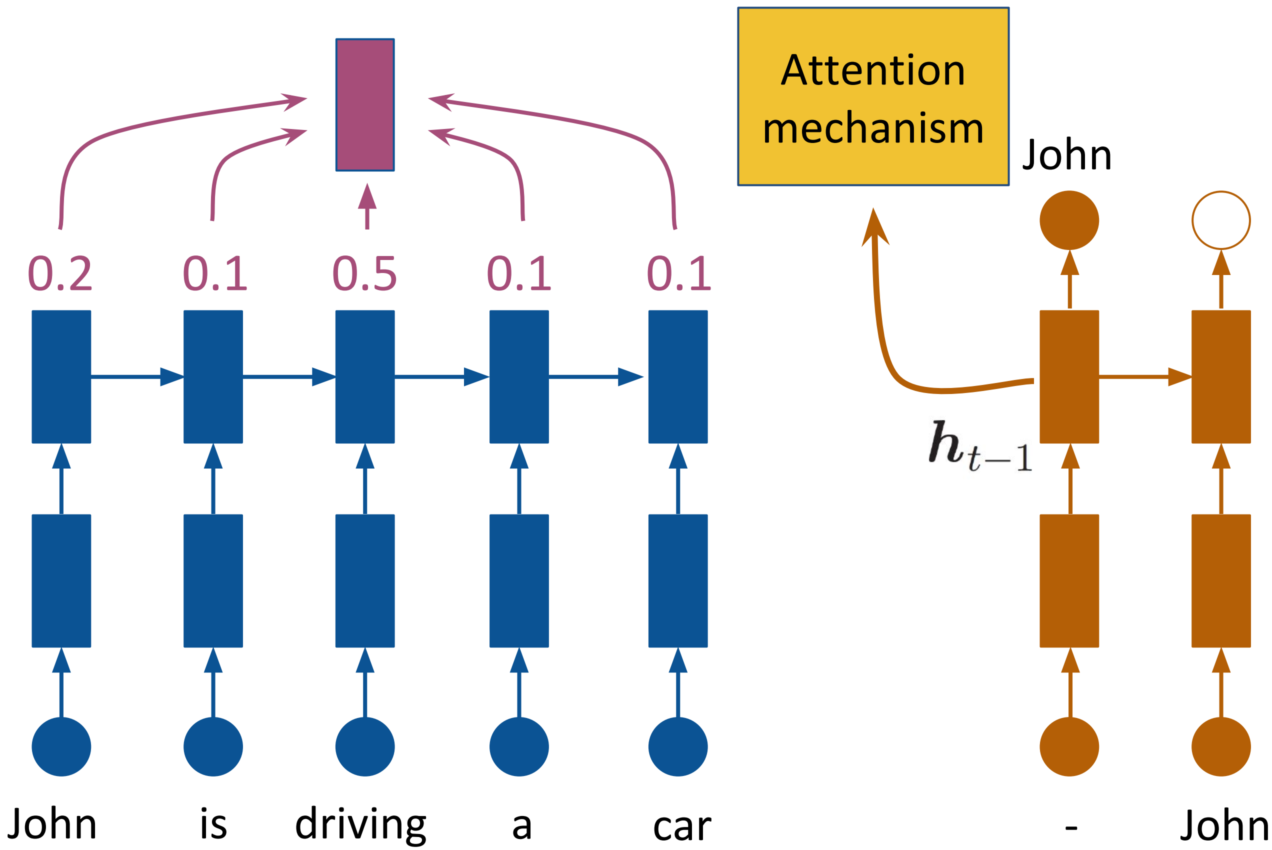
Hassan Sajjad and Fahim Dalvi

Qatar Computing Research Institute, HBKU

Recap: Encoder-Decoder Model



Recap: Attention Mechanism



Today

- Practical considerations
  - Bidirectional LSTMs
  - Dropouts
  - Ensembles
  - Residual Connections
  - Vocabulary limitation in NMT
- Using monolingual data in NMT framework
- Best practices
  - Initialization
  - Batch size
  - Padding
- NMT toolkits

Practical Considerations

Bidirectional LSTM’s

Bidirectional LSTM

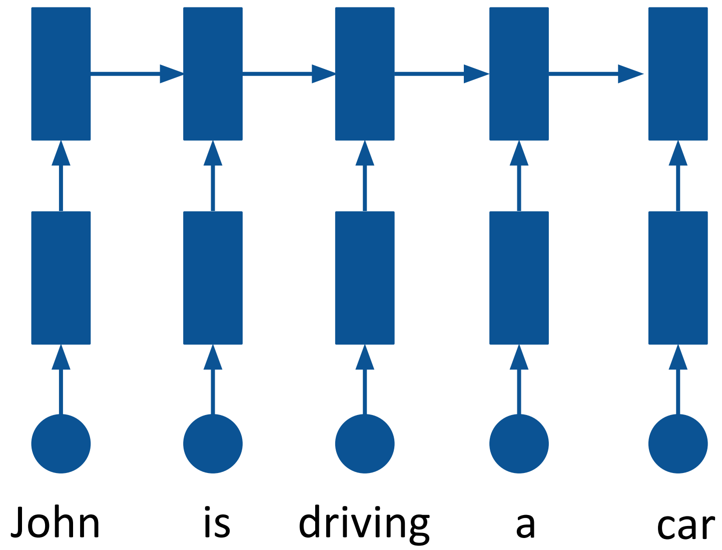
- Currently, our LSTM language model remembers words from the left, but we may need to know some forward context as well to make good decisions

Bidirectional LSTM

- Currently, our LSTM language model remembers words from the left, but we may need to know some forward context as well to make good decisions
- Let’s build two LSTM layers - one that summarizes a sentence from left to right, and another that summarizes a sentence from right to left

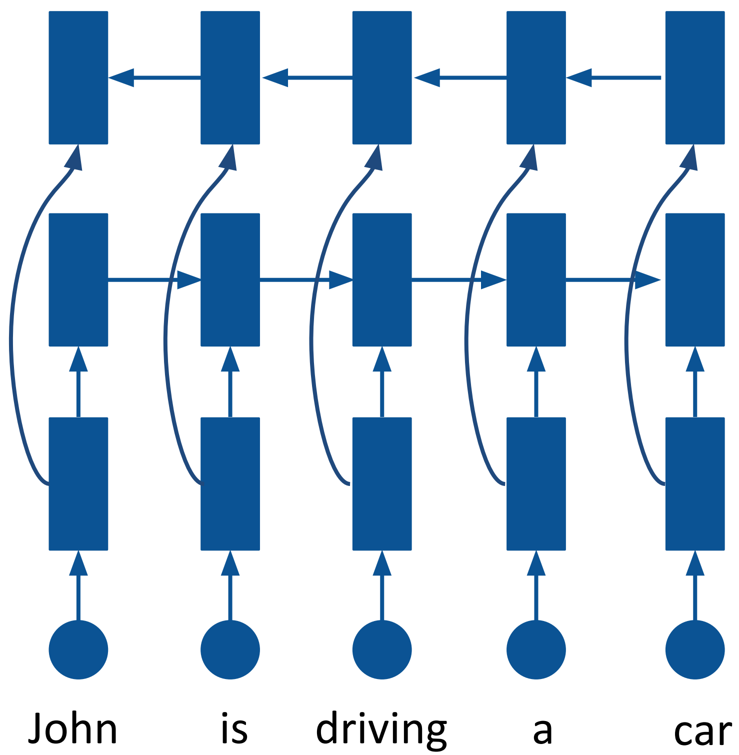
Bidirectional LSTM

Single layer neural machine translation with bidirectional LSTM



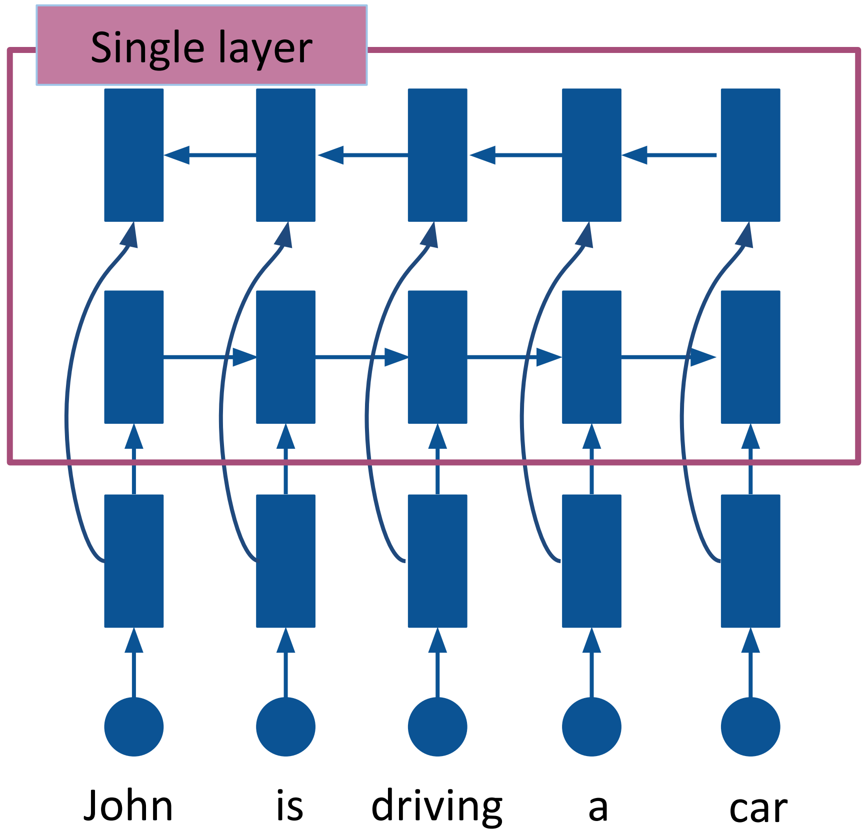
Bidirectional LSTM

Single layer neural machine translation with bidirectional LSTM



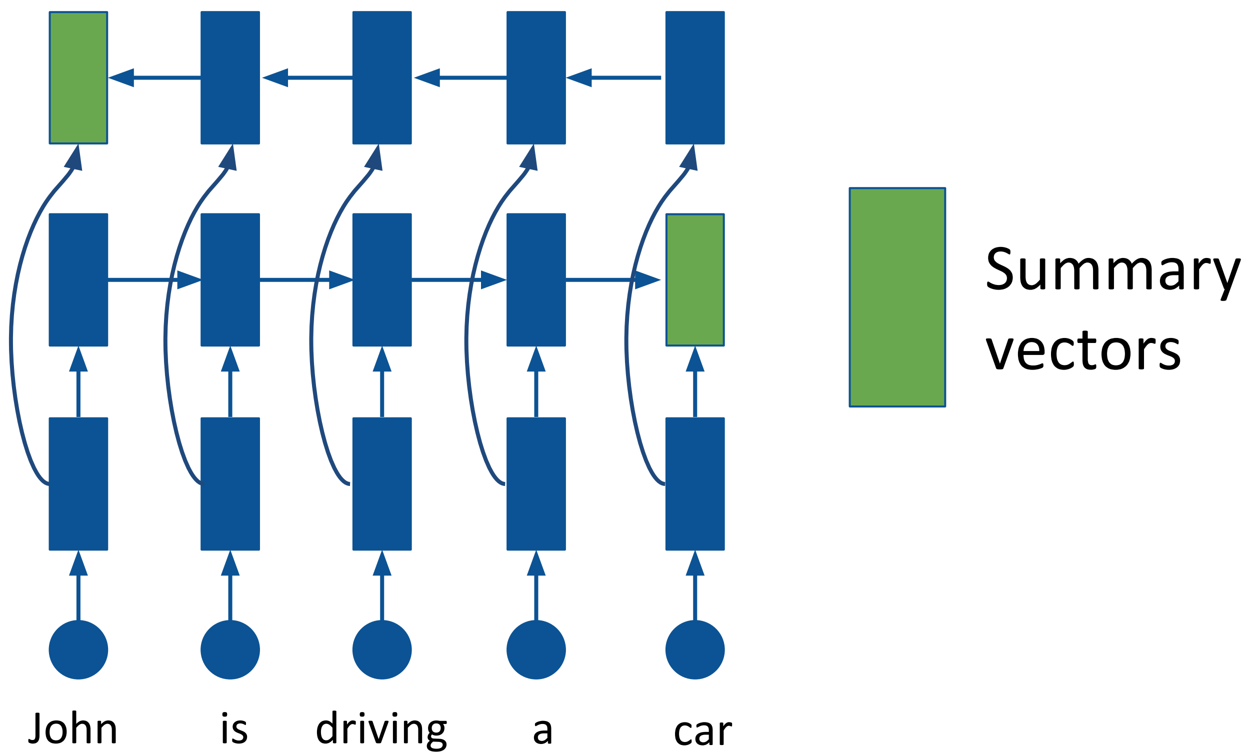
Bidirectional LSTM

Single layer neural machine translation with bidirectional LSTM



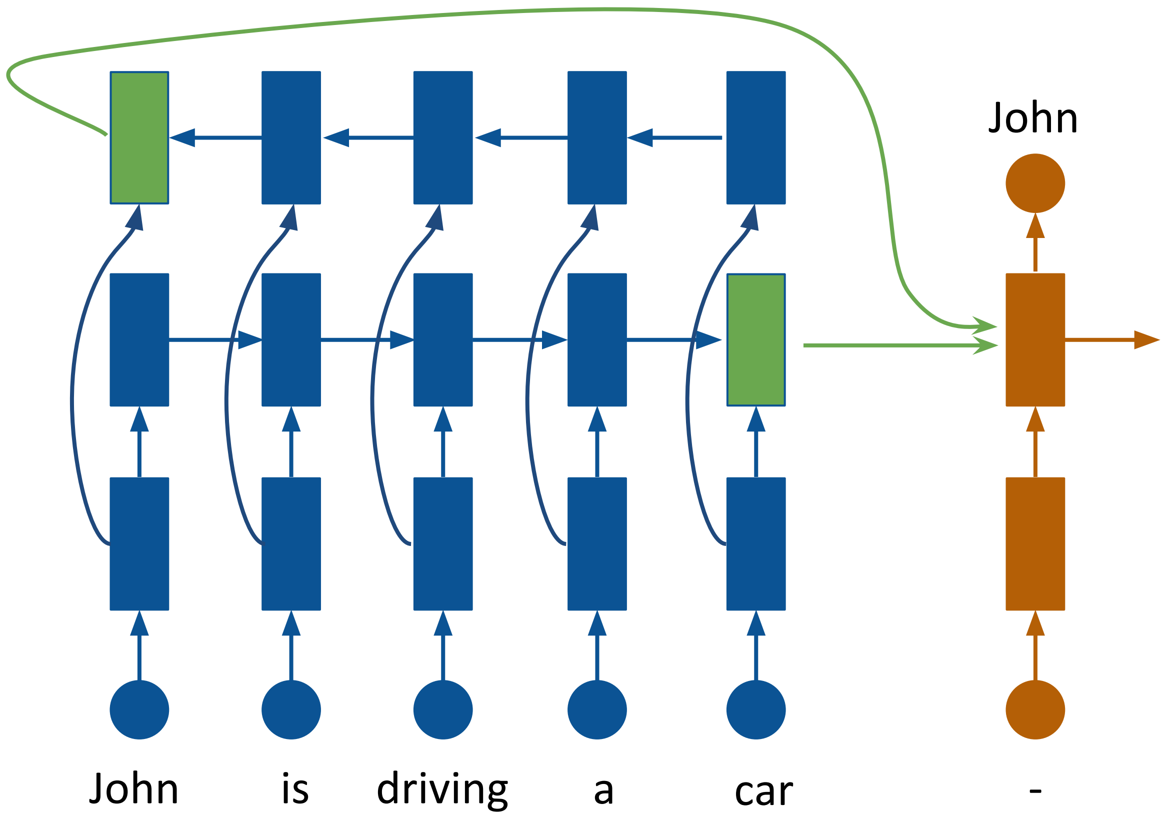
Bidirectional LSTM

Single layer neural machine translation with bidirectional LSTM



Bidirectional LSTM

Single layer neural machine translation with bidirectional LSTM



Both summary vectors are pass to the decoder for prediction

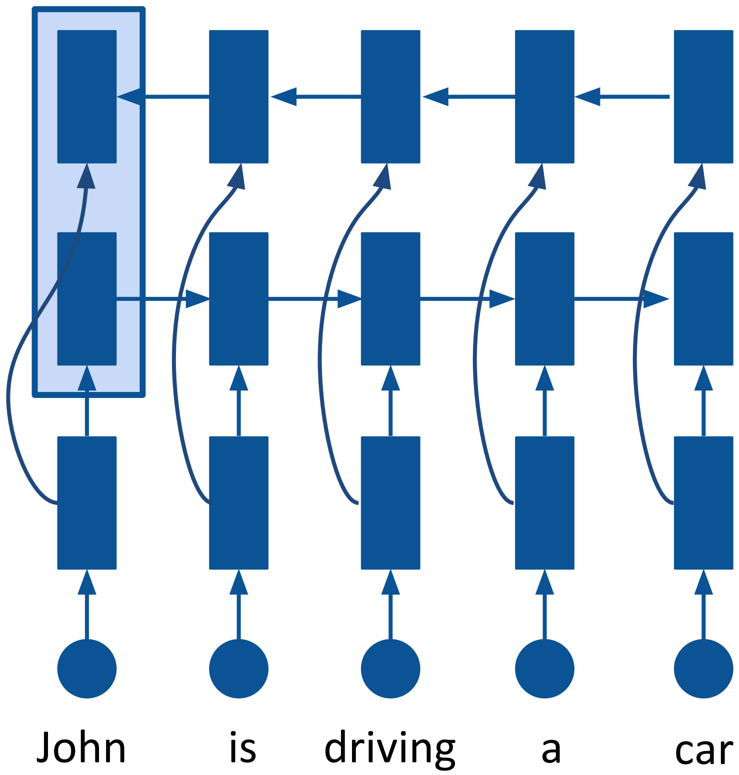
The vectors are concatenated to form a single hidden vector. Vector average is also used occasionally

Practical Considerations

Bidirectional LSTM’s w/ Attention

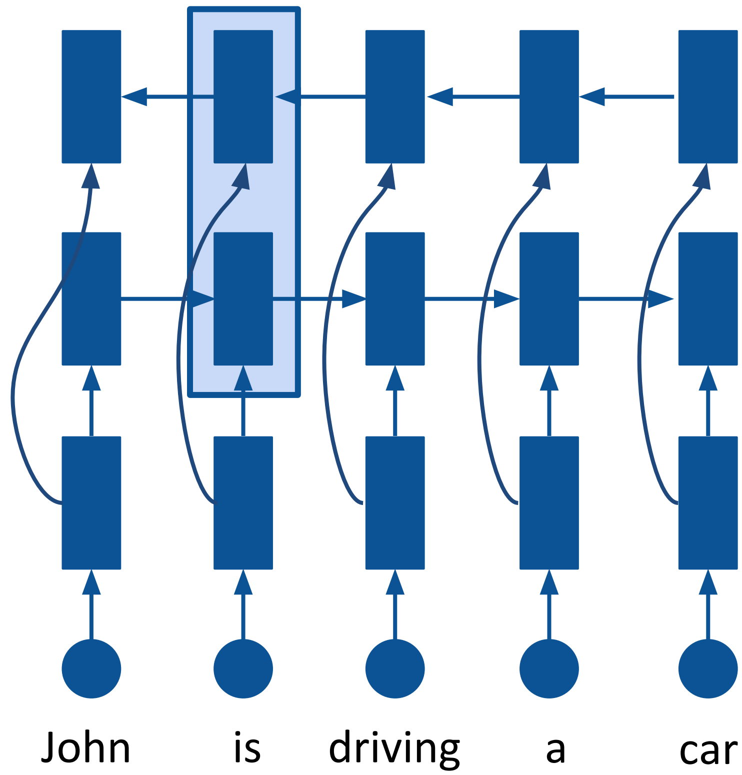
Bidirectional LSTM with Attention

Every state represents a summary of left and right words



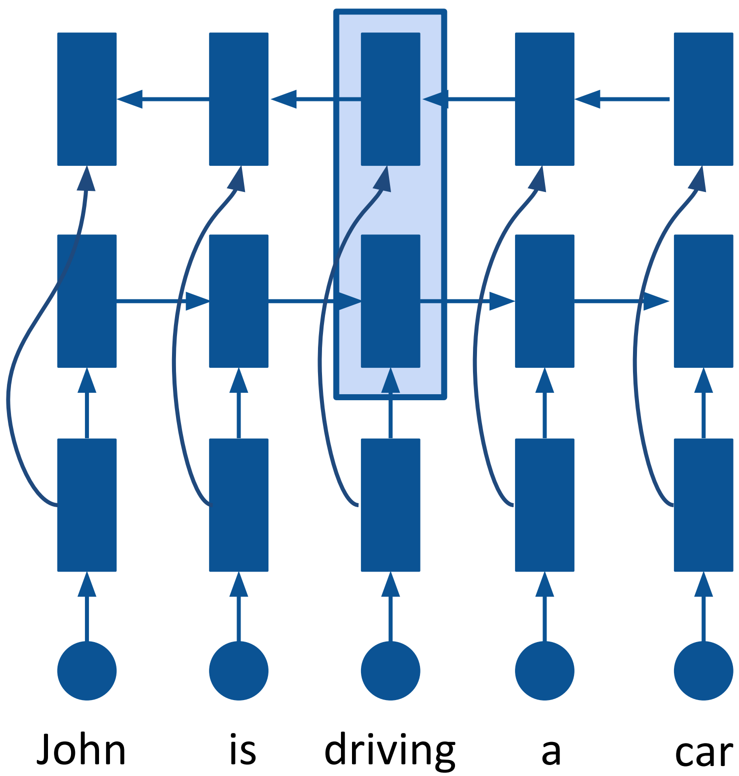
Bidirectional LSTM with Attention

Every state represents a summary of left and right words



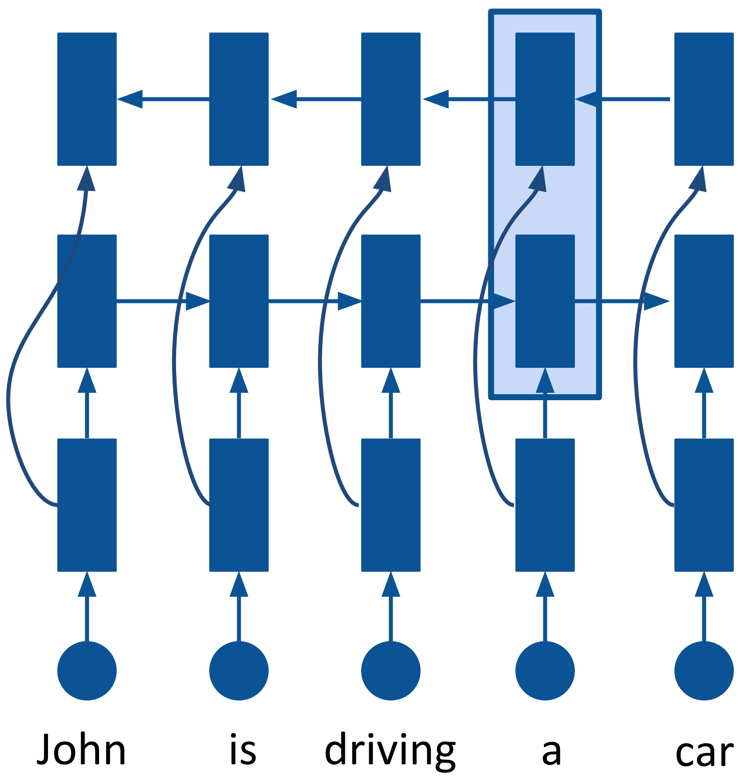
Bidirectional LSTM with Attention

Every state represents a summary of left and right words



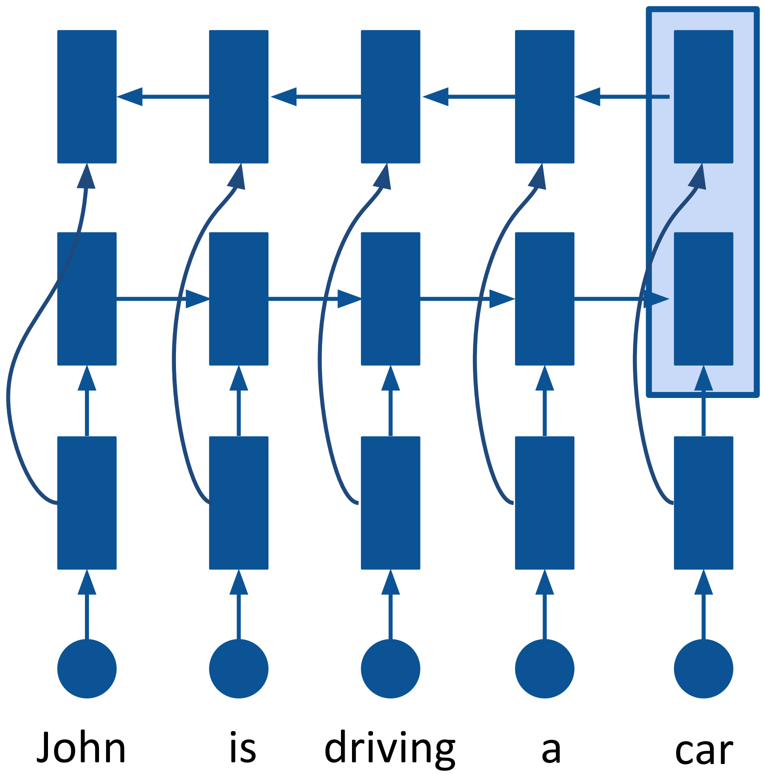
Bidirectional LSTM with Attention

Every state represents a summary of left and right words



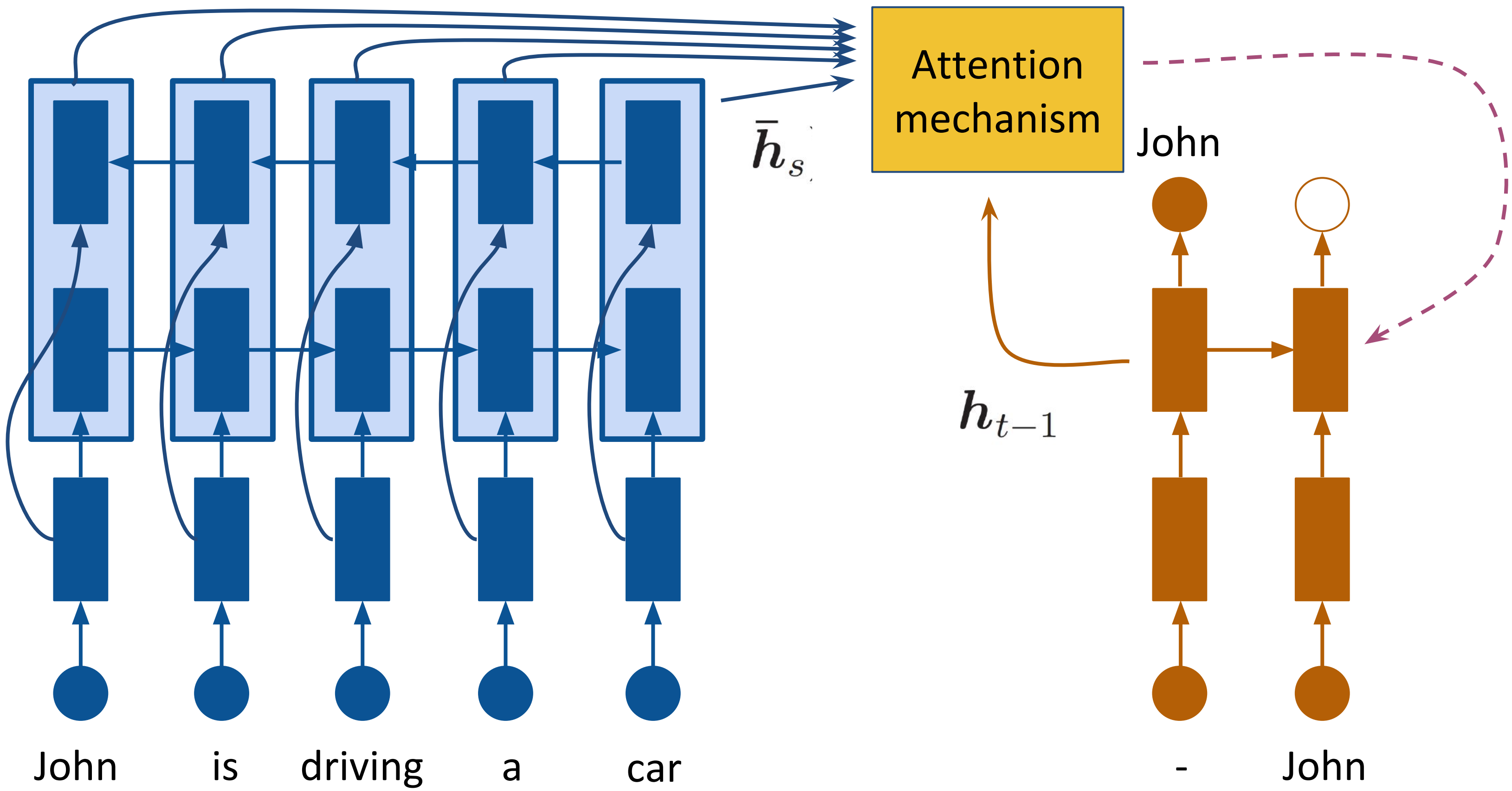
Bidirectional LSTM with Attention

Every state represents a summary of left and right words



Bidirectional LSTM with Attention

Every state represents a summary of left and right words



Bidirectional LSTM with Attention

- Frequently used in competition grade systems
- Results in a 1-2 BLEU points improvements
- A bidirectional LSTM takes 2X more time to train than a unidirectional LSTM

Practical Considerations

Dropout

Overfitting

- Humans tend to believe what comes frequently in their lives or daily routine
  - a recurrent advertisement make us believe it
  - in return, we are less open to any new information

- Similarly, machines tend to overfit what they have seen during training
  - result in less robust model to any unseen scenario

Dropout

- Neural network models tend to overfit the training data
  - performs poorly on unseen data

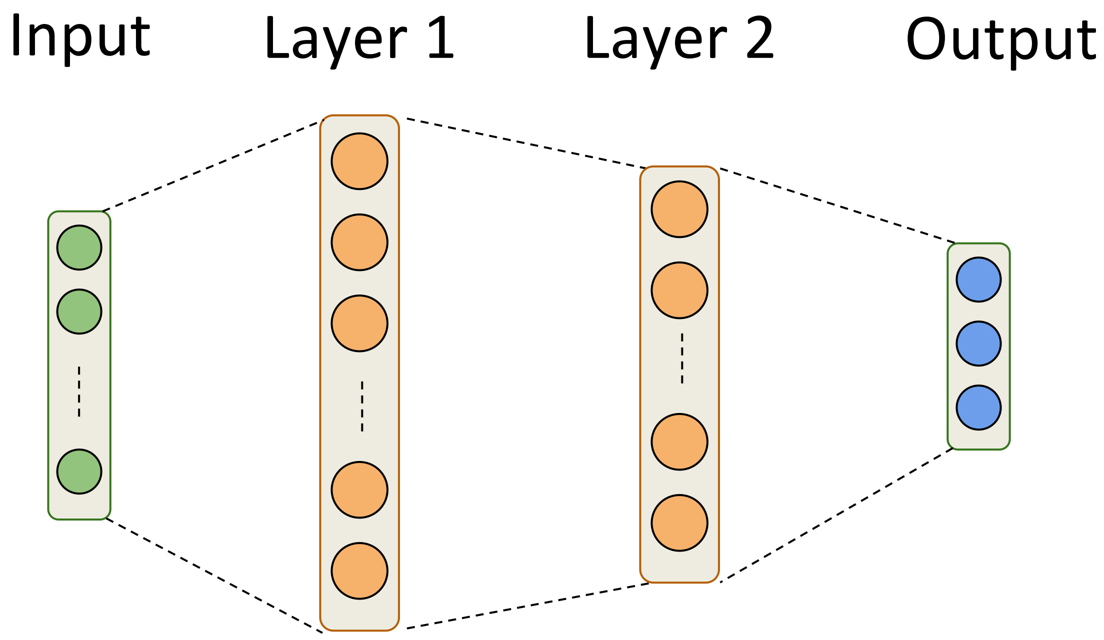
- How can we expose our network to diverse scenarios?
- Can we make the training of the model more robust and improve generalization?

Dropout

"Randomly drop neurons from the network during training"

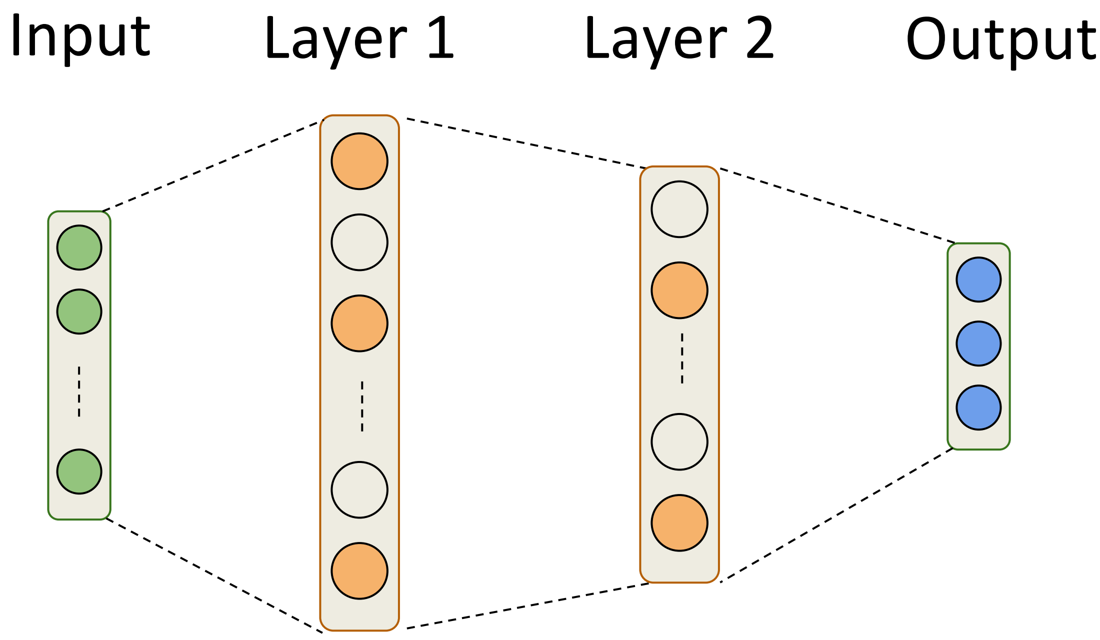
- **Intuition:** give network a wider class of scenarios to tackle
- By dropping a few neurons, we are essentially forcing the model to learn in scenarios when some information is missing
- Some other neuron(s) have to step up to handle this situation

Dropout



- Let's say we want to randomly drop a few neurons from every layer

Dropout



- In other words, some information in the network is missing
- Now model has to learn to reduce loss with fewer neurons
- This improves generalization of the model

Dropout

- Algorithmically, say we want to apply dropout of `p=0.5` on the complete network

- At train time, for every layer in every iteration:
  - For every neuron
    - predict a random number between `0` and `1`
    - if number is less than `0.5`, drop that neuron and its connections

- At test time, **always use the complete network**
  - compensate for missing activations during training by reducing all activations by the factor p

Dropout

- Frequently used in MT specially when training data is small
  - gives a BLEU gain of up to 3 points on small data

Dropout

Implementing in Keras

```
model = Sequential()
model.add(Dense(100, input_shape=(len(x_train[0]),), activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(50, activation='sigmoid'))
model.add(Dropout(0.3))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['acc'])

model.fit(x_train, y_train, verbose=True, epochs=5, validation_split=0.05)
```

Dropout

Implementing in Keras

```
model = Sequential()
model.add(Dense(100, input_shape=(len(x_train[0]),), activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(50, activation='sigmoid'))
model.add(Dropout(0.3))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['acc'])

model.fit(x_train, y_train, verbose=True, epochs=5, validation_split=0.05)
```

Practical Considerations

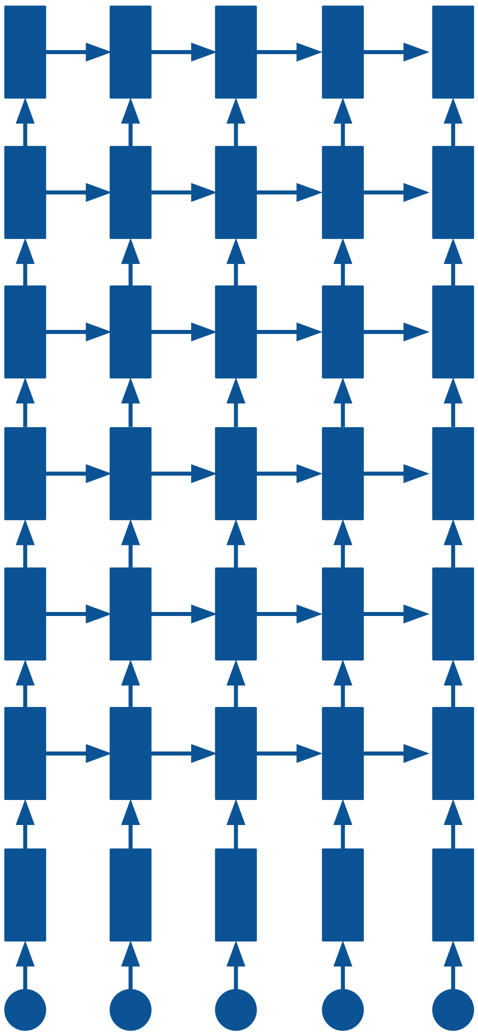
Residual Connections

Residual Connections

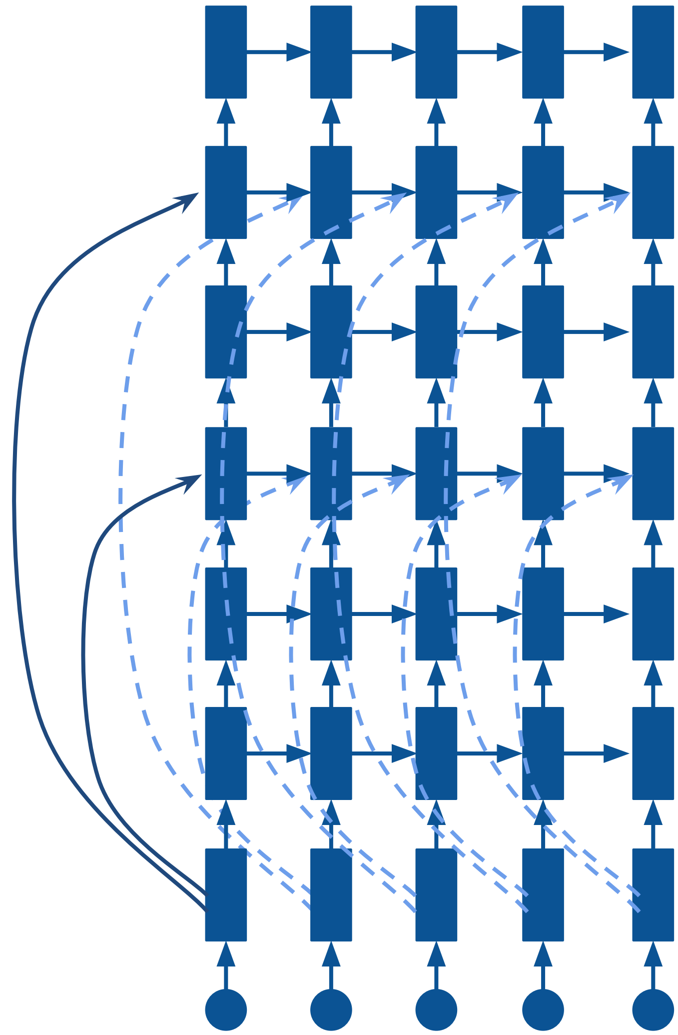
Deeper the better

- As we go deeper in the network:
  - model starts forgetting the input to the network
  - gradient from higher layers to initial layers starts diminishing (vanishing gradient)
- Residual networks keep shortcut connections between different layers
  - in practice, several combinations of connections are used between layers

Residual Connections



Residual Connections



Shortcut connections from the embedding layer to higher layers is one common implementation

Note that we are increasing the overall number of parameters in the network!

Practical Considerations

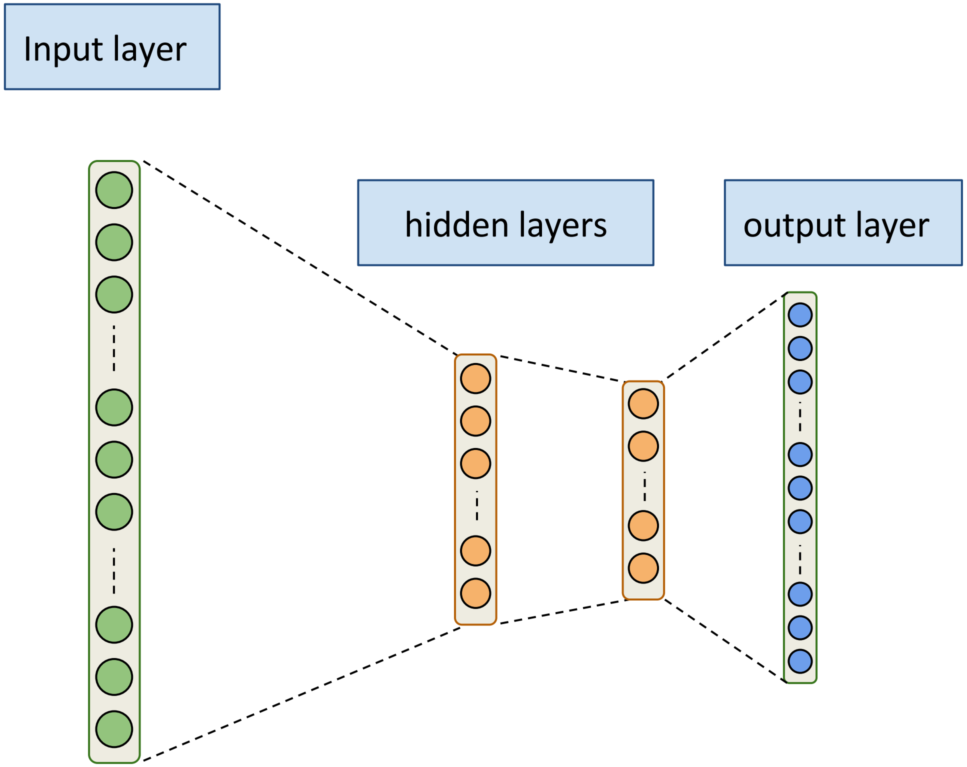
Ensembles

Ensemble

- Neural network models are trained on random weights, thus result in different models
- Every model may have specialized in slightly different aspects of the data
- In ensemble, we combine several trained models at test time

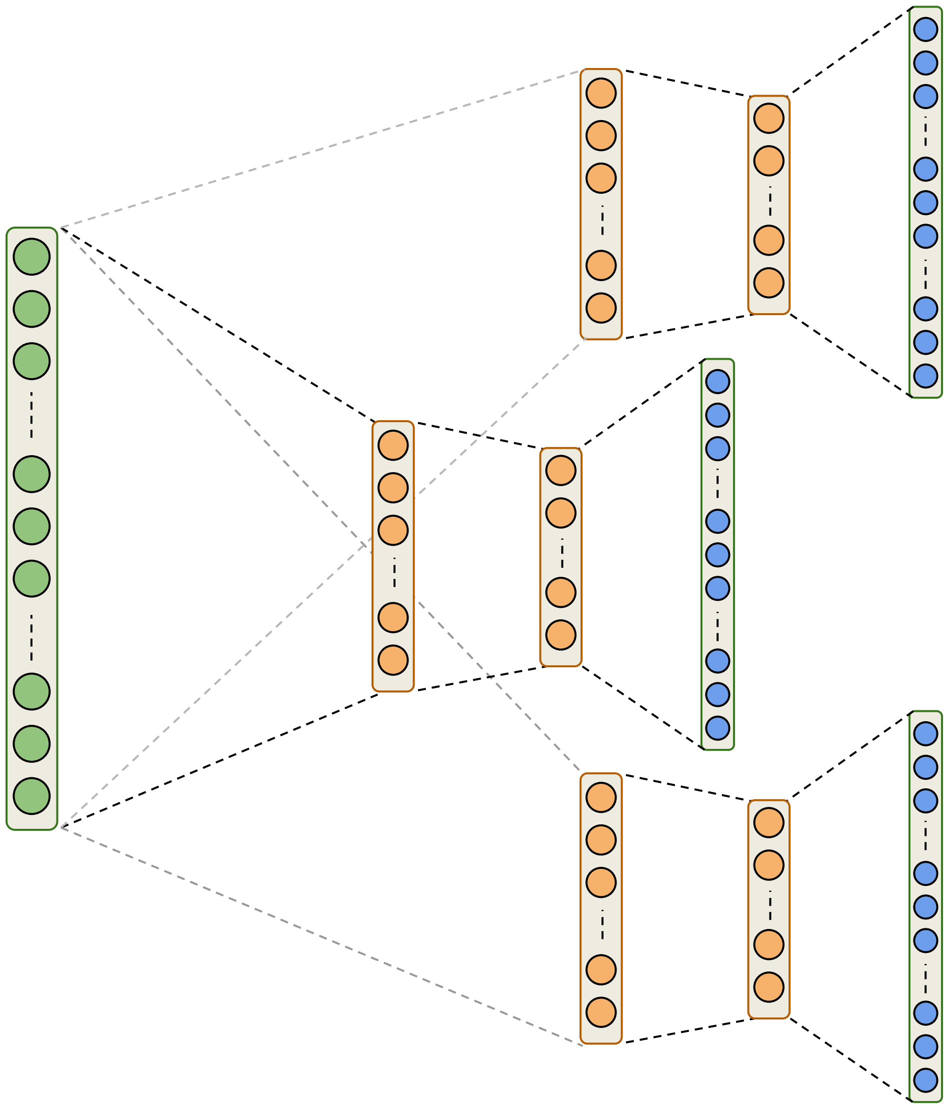
Ensemble

Single model



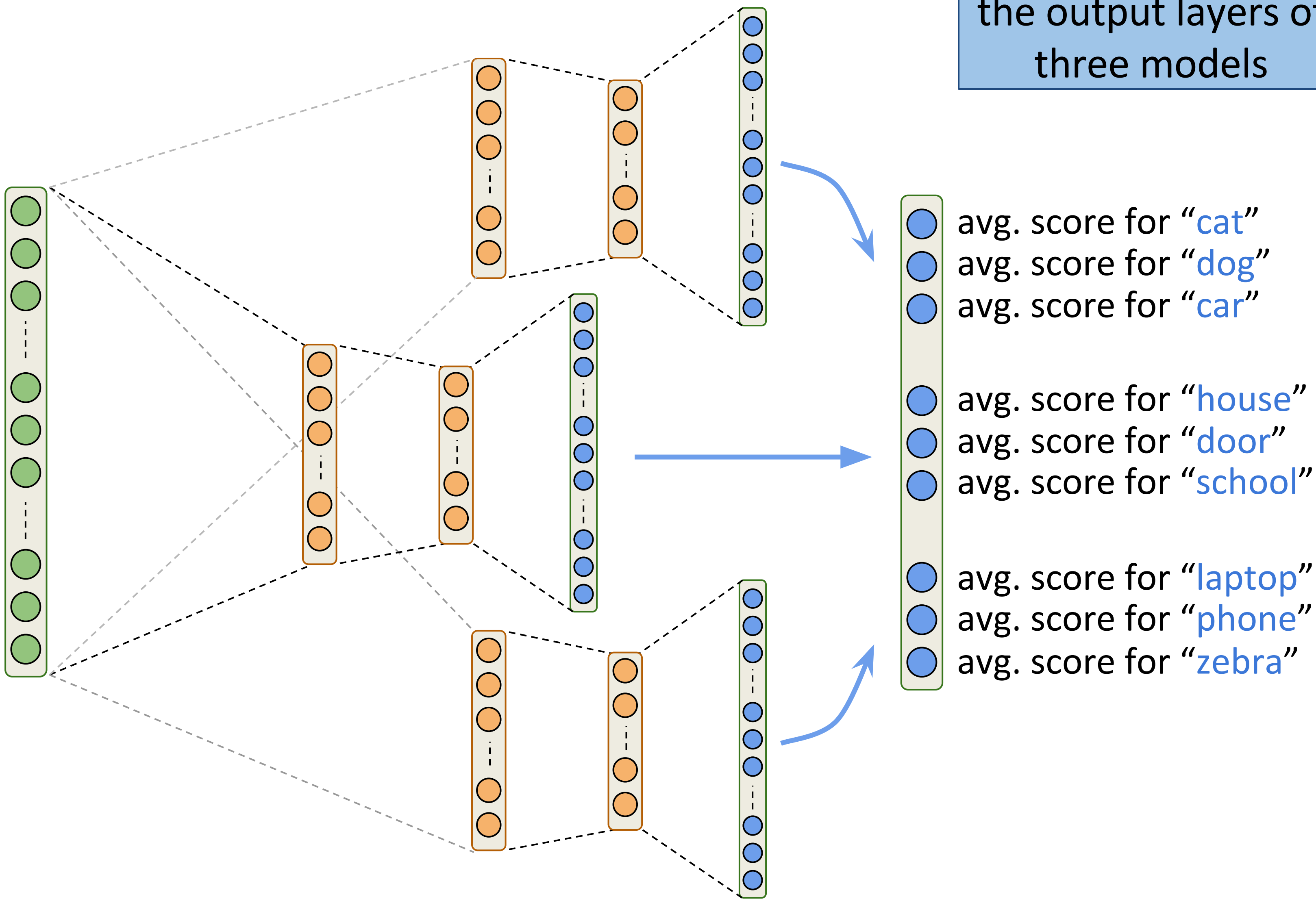
Ensemble

Three models



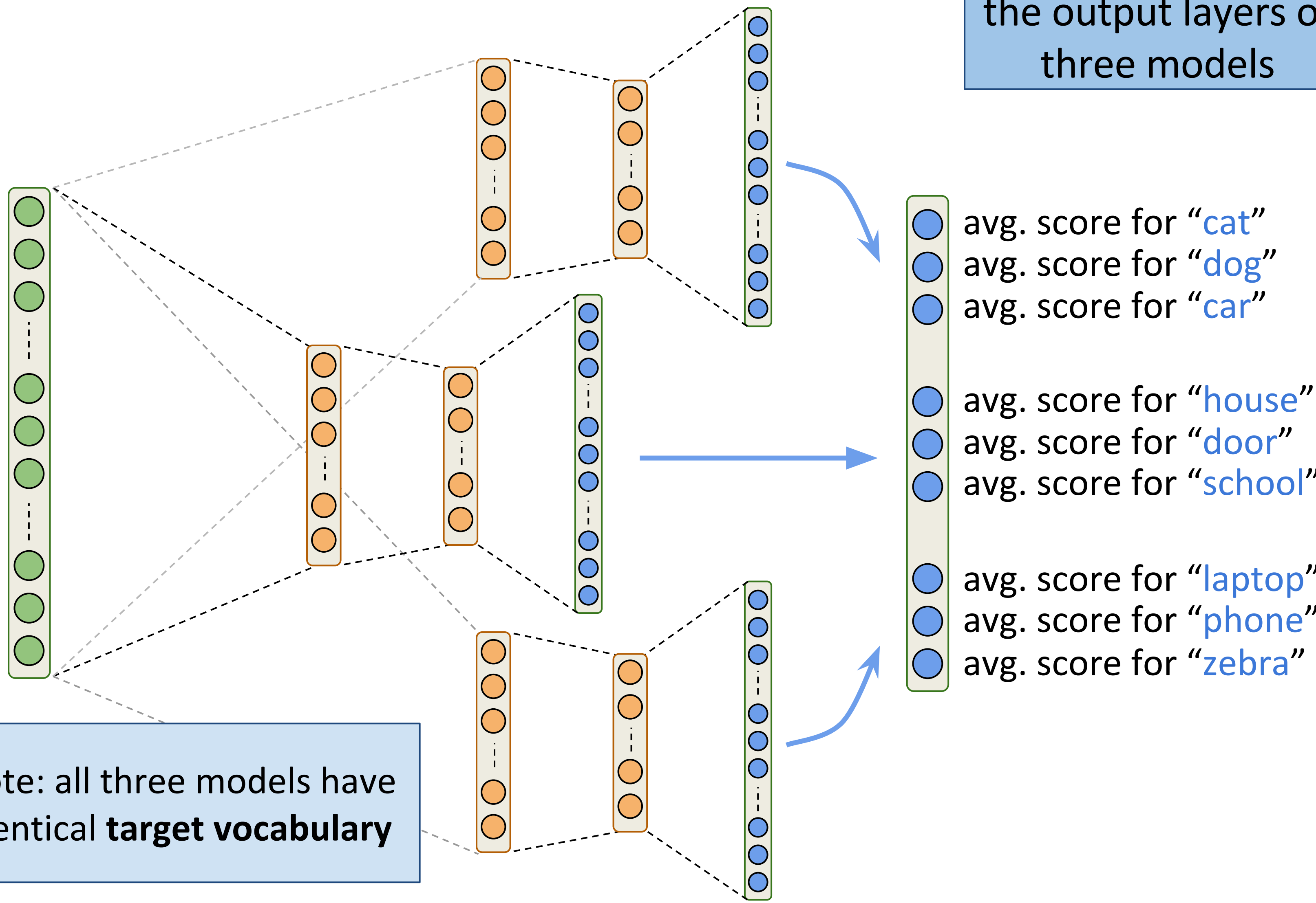
Ensemble

Average scores of the output layers of three models



Ensemble

Average scores of the output layers of three models



Note: all three models have identical **target vocabulary**

Ensemble

- Ensemble gives a performance improvement of 1-2 BLEU points
- Slow for real time processing
- Ensemble can also be done on models trained on different datasets but share identical target vocabulary
  - beneficial to combine models trained on different data

Practical Considerations

Vocabulary size limitations

Vocabulary Size Limitation

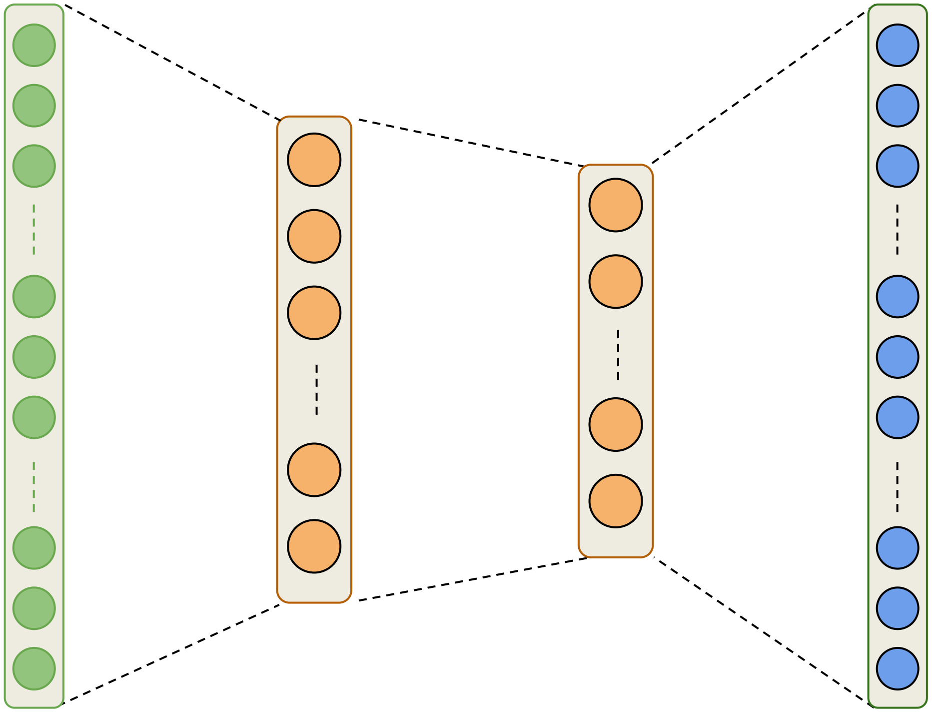
- Languages have unlimited vocabulary
  - increases every day
  - morphologically rich languages have much richer vocabulary

Fallschirm	(Parachute)
Fallschirmspringer	(Parachute Jumper)
Fallschirmspringerschule	(Parachutist School)

- Training time of neural models is very sensitive to vocabulary size

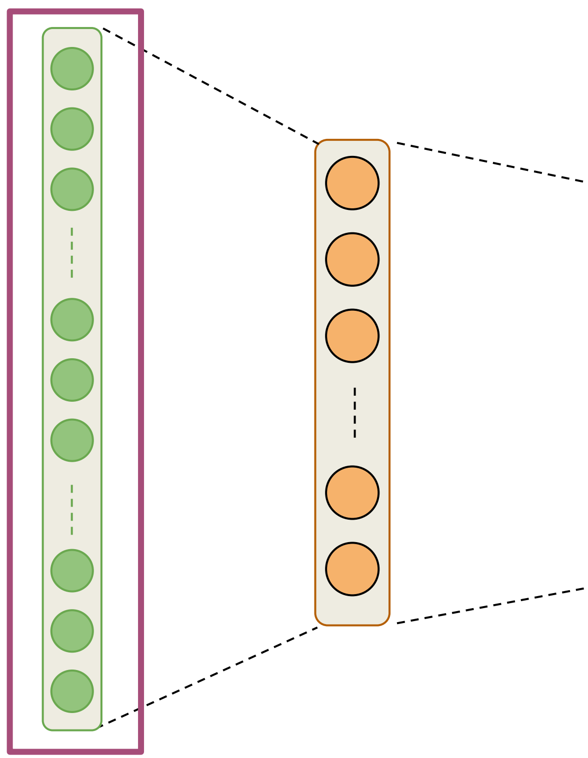
Vocabulary Size Limitation

- Training time of neural models is very sensitive to vocabulary size



Vocabulary Size Limitation

- Training time of neural models is very sensitive to vocabulary size



Larger source side vocabulary means a longer one-hot vector!

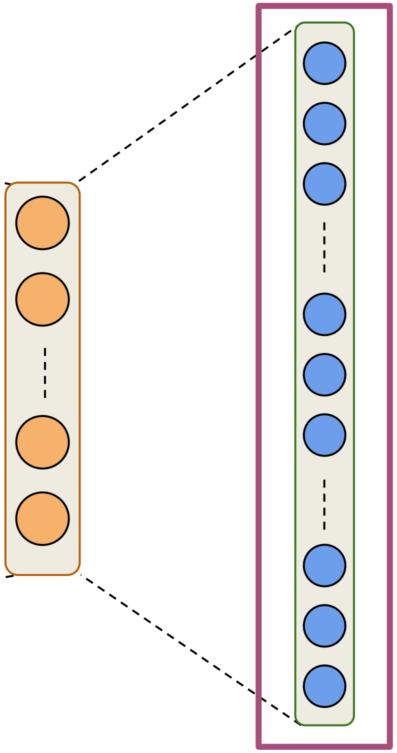
Number of parameters increases, and so does computation time for the associated multiplications

Vocabulary Size Limitation

- Training time of neural models is very sensitive to vocabulary size

Larger target side vocabulary means a lot more scores and probabilities to compute!

Softmax is notoriously slow for large vectors, slowing down the entire training process



Vocabulary Size Limitation

- **Word-based** models suffer from data sparseness

Play Playing Played

- The above words are represented as three vocabulary units, when they mean very similar things

Vocabulary Size Limitation

- **Word-based** models suffer from data sparseness

Play Playing Played

- The above words are represented as three vocabulary units, when they mean very similar things
- Explicit or implicit handling of various language phenomena can reduce data sparseness

Vocabulary Size Limitation

- A typical solution is to **choose most frequent source and target vocabulary words** and replace infrequent words with a unique token, say <UNK>

Vocabulary Size Limitation

- A typical solution is to **choose most frequent source and target vocabulary words** and replace infrequent words with a unique token, say <UNK>
- The **downside** of this method is the increase in the number of unknowns during testing
- We also have no explicit handling of unknown words that were not present in the training corpus

Vocabulary Size Limitation

- Researchers have proposed several **sub-word** based methods to handle the problem of vocabulary limitation and unknown words handling

Words

Common Sub-word

Play Playing Played

Play

Subword Modeling

Word representations:

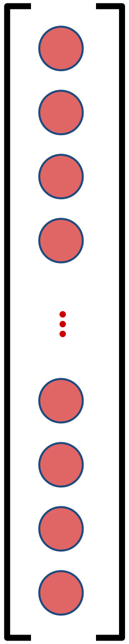
- Character based
- Character-LSTM based
- Character-CNN based
- Frequency based sub-word segmentation (BPE, word piece)
- Hybrid representation

Practical Considerations

Subwords: Character-based

Character Units

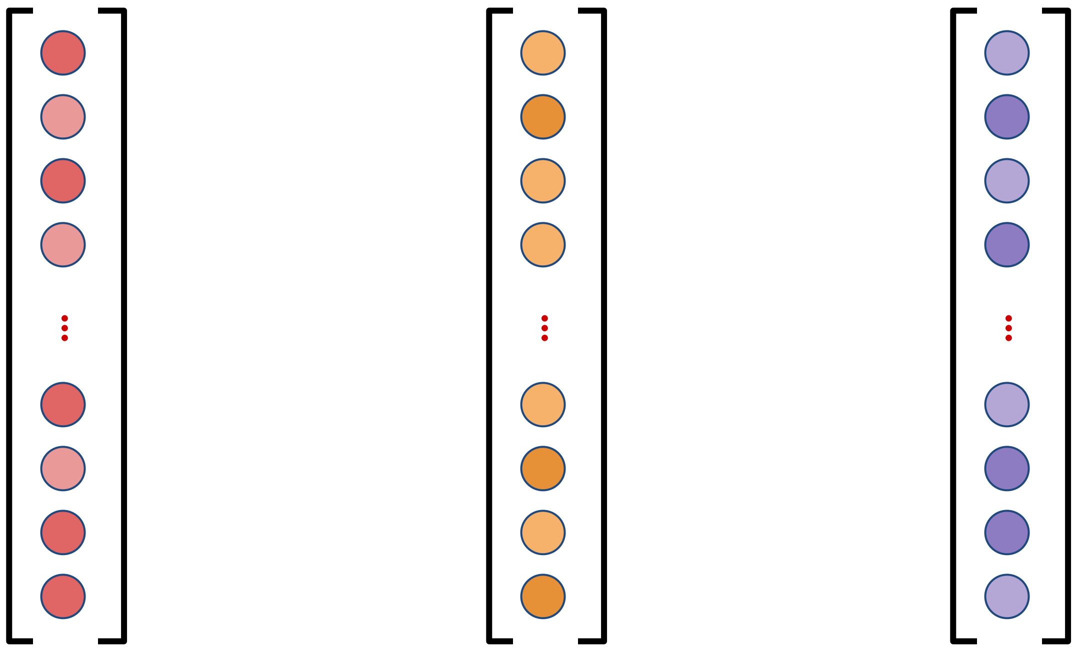
A *word embedding* is represented as a vector per word:



Embedding of
"car"

Character Units

A *character embedding* is represented as a vector per character:



Embedding of "c"   Embedding of "a"   Embedding of "r"

Character Units

Q: How can we use these practically in our **seq2seq** models?

Character Units

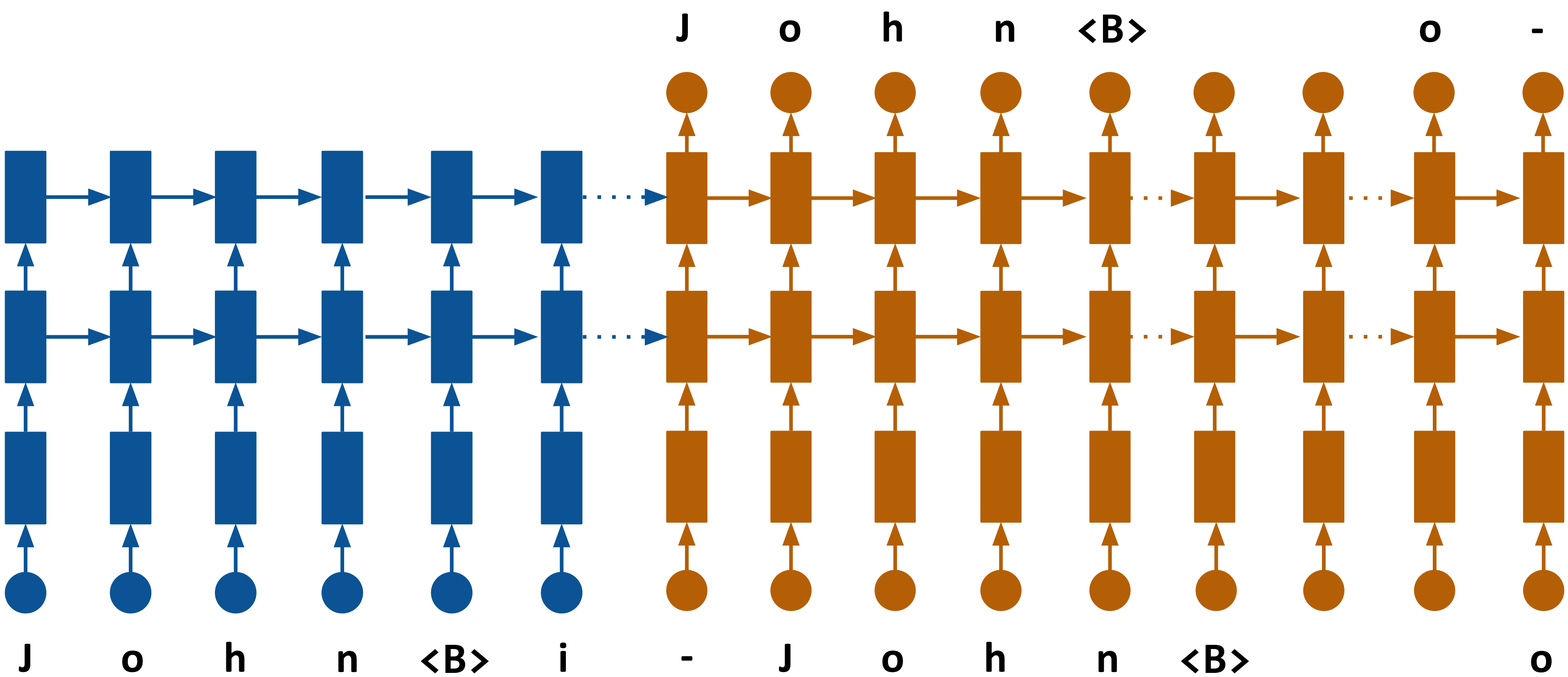
Q: How can we use these practically in our **seq2seq** models?

A: Just split the words into characters, and use a special symbol to keep track of word boundaries

```
J o h n <B> i s <B> d r i v i n g <B>
```

Keep everything else the same, our vocabulary will now be all characters and the special symbol!

Character Units



Character Units

- Pros
  - Vocabulary size is just the **total unique characters** in the corpus
  - Model learns to handle morphological variations
    - play, playing, played
- Cons
  - Conceptually, characters do not represent a semantic unit in contrast to words
  - Long sequence length
    - makes prediction at train and test time very slow
    - a **large number** of softmax operations (albeit smaller operations)

Character Units

A few combinations to try:

- source characters to target words
- source characters to target characters
- source words to target characters

Practical Considerations

Subwords: Frequency based

Frequency based Subword Units

- Byte Pair Encoding *(most used in current NMT systems)*
  - Sennrich, Haddow, Birch. *Neural Machine Translation of Rare Words with Subword Units.* ACL 2016
- Word Piece Model
  - Wu et al. 2016, *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*

Frequency based Subword Units

- Byte Pair Encoding *(most used in current NMT systems)*
  - Sennrich, Haddow, Birch. *Neural Machine Translation of Rare Words with Subword Units.* ACL 2016

- Word Piece Model
  - Wu et al. 2016, *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*

Byte Pair Encoding

- Frequency-based segmentation
  - split words to reduce vocabulary size
- Words have various overlapping segments

  Play Playing Played Playstation
- Segment them based on frequency

  Play ing ed station
- Essentially, mapping all forms of “Play” to it’s root form

Byte Pair Encoding

Algorithm

- Start from vocabulary equal to total number of characters
- Split words into characters
- Merge most frequent ngrams to form a new pair

Byte Pair Encoding

Start with character vocabulary

Dictionary

5 l o w
2 l o w e r
6 n e w e s t
3 w i d e s t

Vocabulary

l, o, w, e, r, n, s, t, i, d

Example from Sennrich's slides

Byte Pair Encoding

“e s” comes 9 times together

Dictionary

5 l o w
2 l o w e r
6 n e w e s t
3 w i d e s t

Vocabulary

l, o, w, e, r, n, s, t, i, d

Example from Sennrich’s slides

Byte Pair Encoding

"e s" comes 9 times together

Dictionary

5 l o w
2 l o w e r
6 n e w **es** t
3 w i d **es** t

Vocabulary

l, o, w, e, r, n, s, t, i, d, **es**

Example from Sennrich's slides

Byte Pair Encoding

Similarly, "l o" comes 7 times together and "es t" comes 9 times together

Dictionary

5	**lo** w
2	**lo** w e r
6	n e w est
3	w i d est

Vocabulary

l, o, w, e, r, n, s, t, i, d, es, **lo**, **est**

Example from Sennrich's slides

Byte Pair Encoding

Total number of merge operations done = 3

Dictionary

5	**lo** w
2	**lo** w e r
6	n e w est
3	w i d est

Vocabulary

l, o, w, e, r, n, s, t, i, d, **es**, **lo**, **est**

Example from Sennrich's slides

Byte Pair Encoding

Vocabulary size = initial character size + number of merge operations

Dictionary

5	**lo** w
2	**lo** w e r
6	n e w est
3	w i d est

Vocabulary

l, o, w, e, r, n, s, t, i, d, **es**, **lo**, **est**

Example from Sennrich's slides

Byte Pair Encoding

- One hyperparameter, number of operations
  - defines vocabulary size e.g. 50k operations limit vocab to 50k
- Widely used in research and competition grade systems
- Other benefits
  - learns to transliterate
  - morphological variation handling, e.g. drive, driving, driven
  - segmentation of morphologically rich languages

Byte Pair Encoding

- Arabic segmentation
  - segmentation is vital
  - Sajjad, et al. *Challenging Language-Dependent Segmentation for Arabic: An Application to Machine Translation and Part-of-Speech Tagging.* ACL 2016
  - not morphological segmentation but works for extrinsic tasks

Byte Pair Encoding

- Downside
  - segmentation is not morphologically consistent
    - e.g. drive, driving, driven may have different segmentations
    - can not benefit from same root forms
  - unknown word segmentations may lead to semantically different translations
    - e.g. greenhouse -> green house
  - segmentation is learned independent of the translation quality

Standard Practices

- For languages with similar writing scripts like German, English, French
  - combine the training data and train a BPE model
  - WMT system use 89,900 BPE operations
- For languages with different writing scripts like English, Arabic, Hebrew
  - build BPE models separately for each language
  - IWSLT English-Arabic systems use 50,000 BPE operations for both source and target languages

Practical Considerations

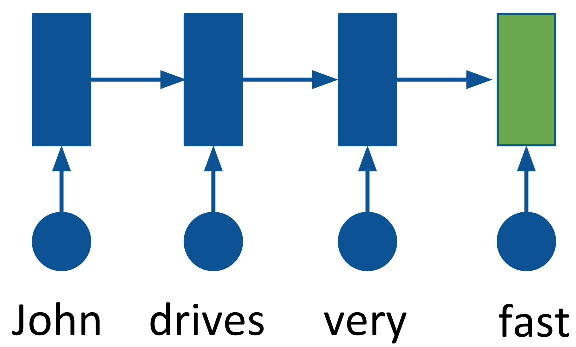
Subwords: Character-LSTM based

Character LSTM

We can learn word embeddings based on character LSTMs

Character LSTM

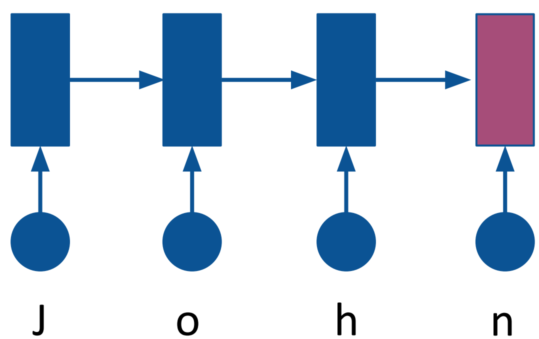
We can learn word embeddings based on character LSTMs



Summary vector with summary of all the **words**

Character LSTM

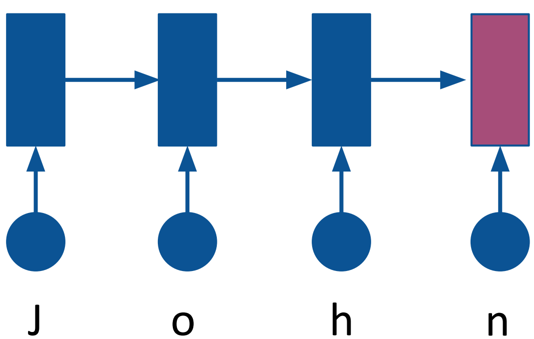
We can learn word embeddings based on character LSTMs



Summary vector with summary of all the **characters**

Character LSTM

We can learn word embeddings based on character LSTMs

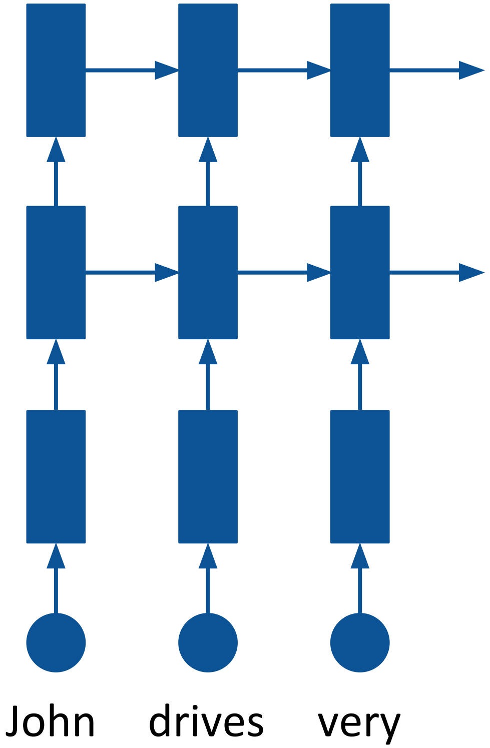


Summary vector with summary of all the **characters**

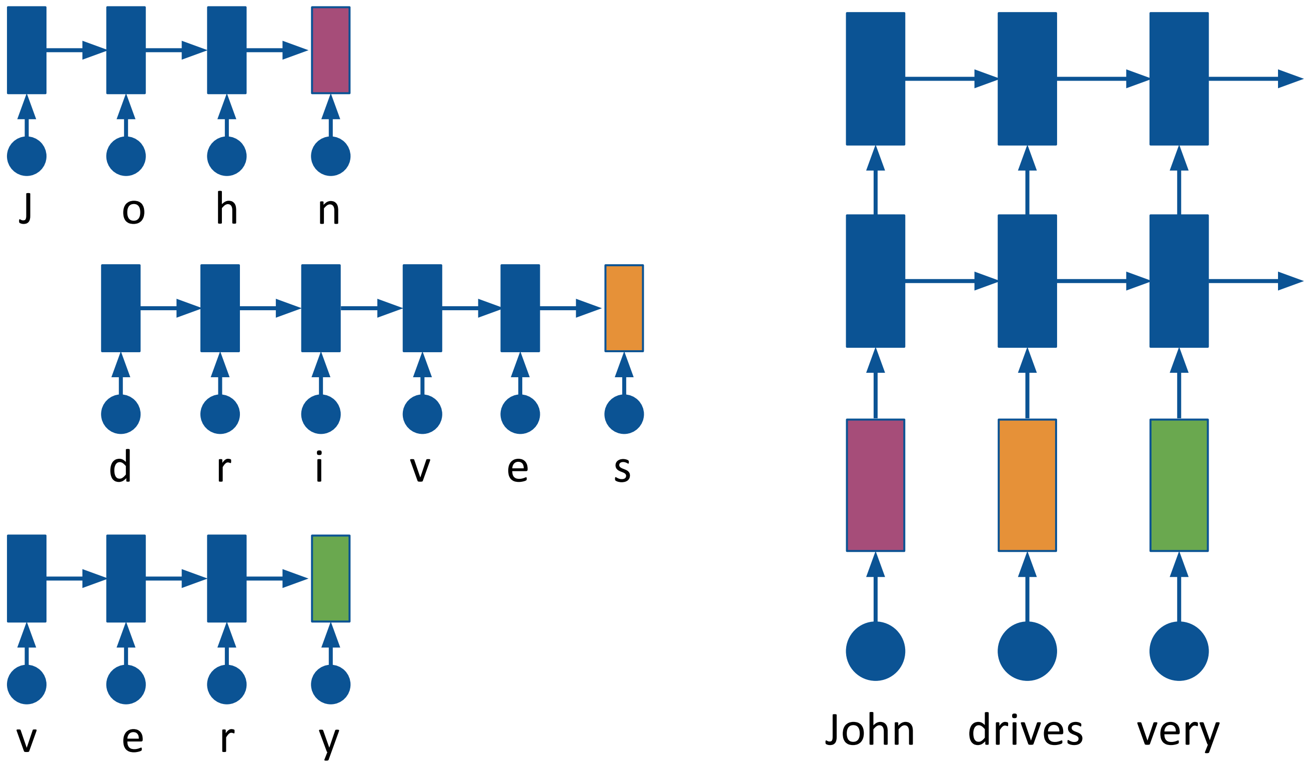
We can also consider this summary as an embedding of “John”

Character LSTM

Usually, our first layer is a feed-forward layer which we treat as embeddings



Character LSTM

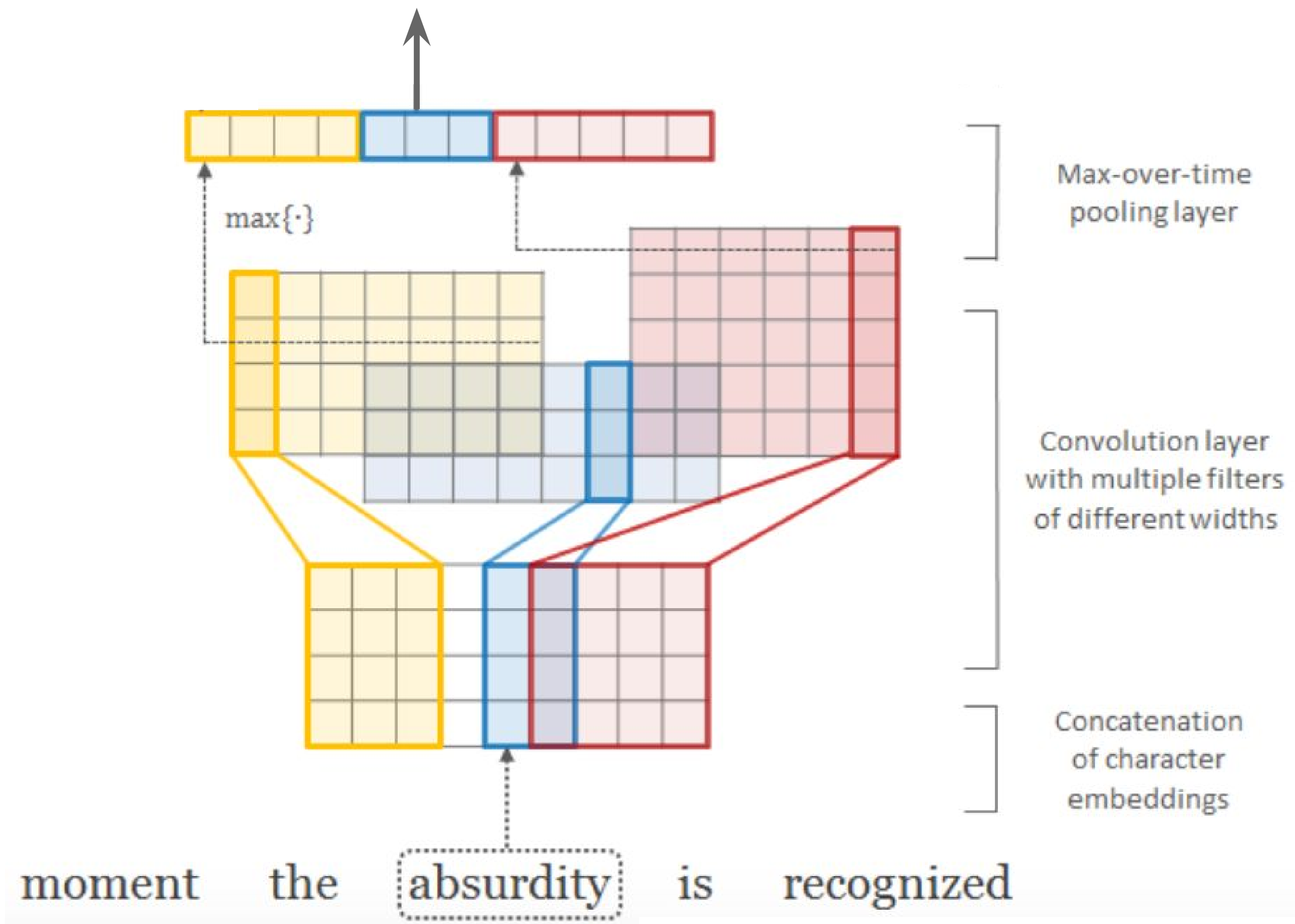


In this case, we will use our computed embeddings in our encoder or decoder

Practical Considerations

Subwords: Character-CNN based

Character CNN



Kim, Jernite, Sontag, Rush, *Character-Aware Neural Language Model*, AAAI 2016

Character LSTM & CNN

Pros

- Word representations using characters are richer than word-based embeddings
  - incorporate morphological information
    - play, playing, played, etc.
- Works well in translating morphologically rich languages
- No issue of vocabulary size
  - vocabulary is equal to total number of characters

Character LSTM & CNN

Cons

- Significantly slower than normal feed-forward layers
- Number of parameters is higher

Practical Considerations

Subwords: Hybrid Units

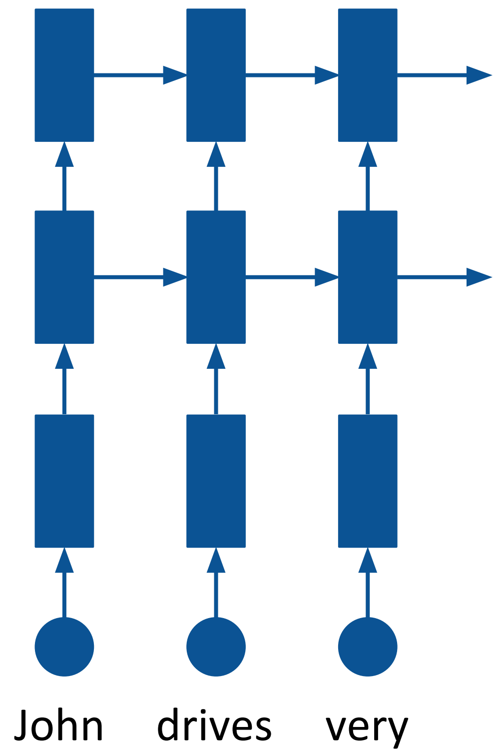
Hybrid Units

Intuition: Use word level model most of the time, but consult character model for unknown words

Convert infrequent words to characters for training

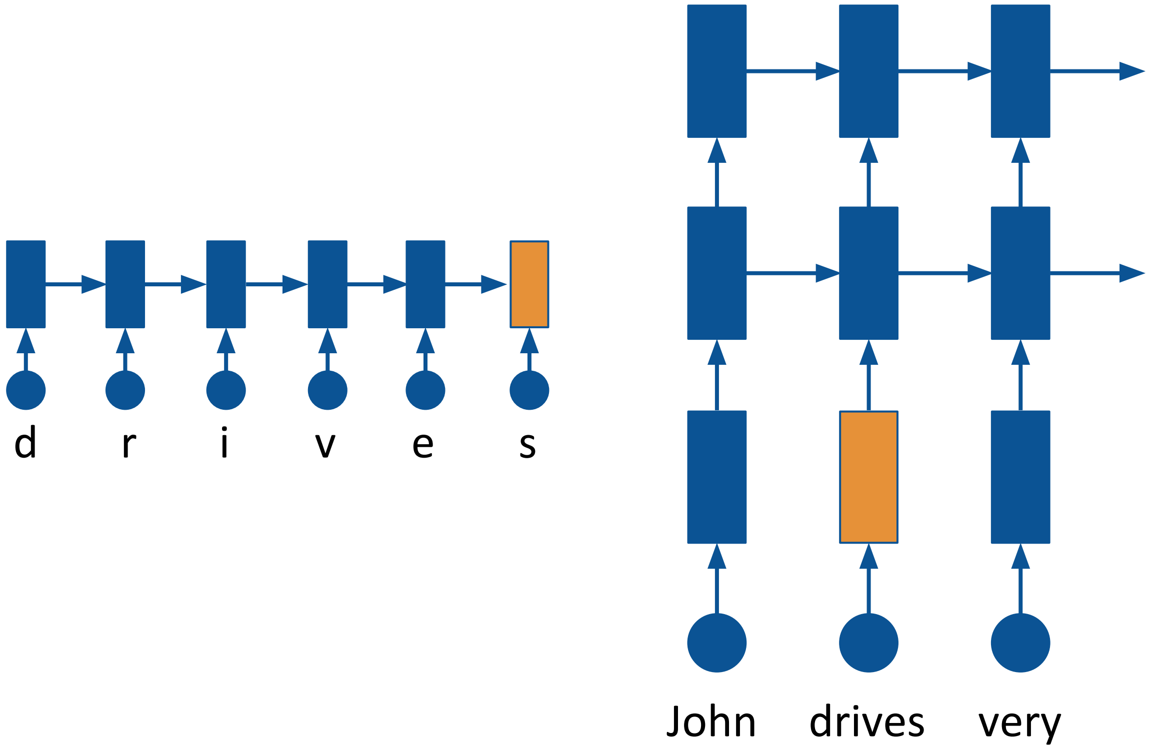
Luong, Manning, *Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models,* ACL 2016

Hybrid Units: Source side



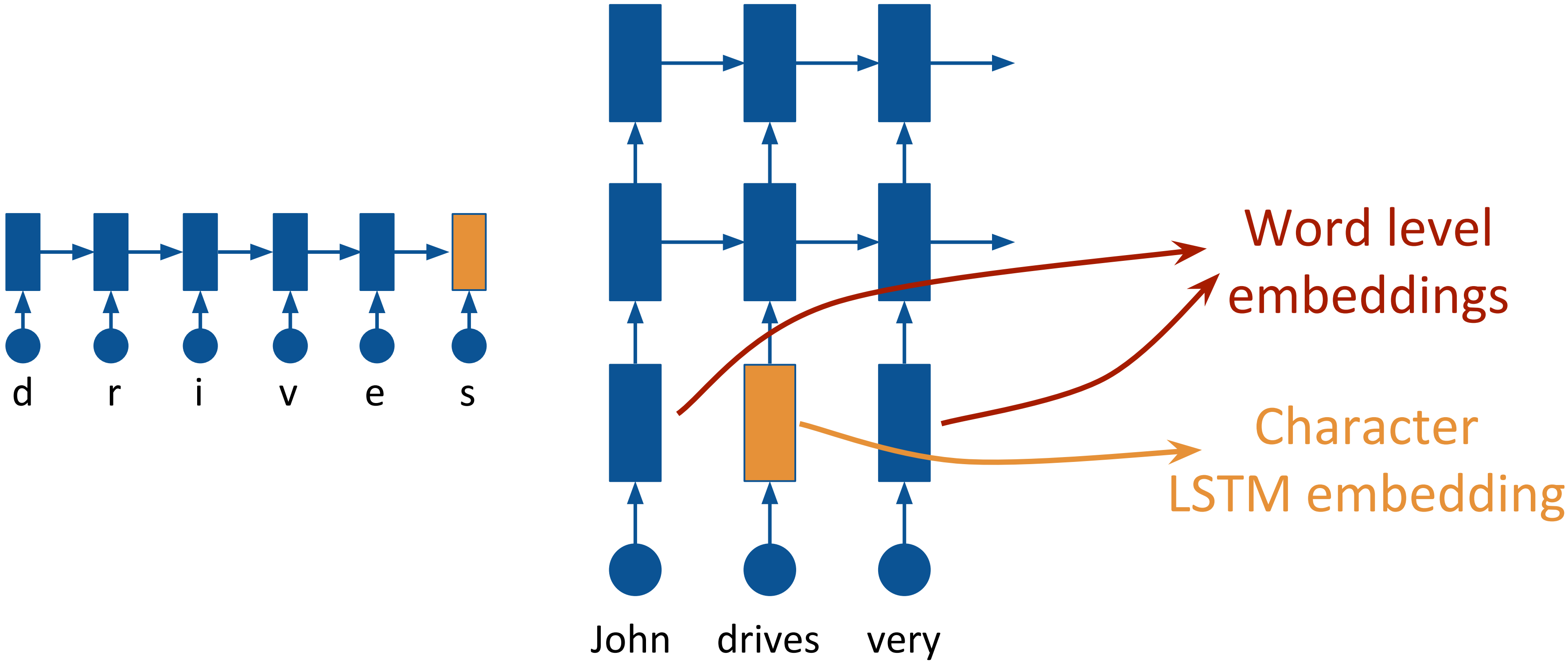
Consider the case where “**drives**” is an unknown word

Hybrid Units: Source side



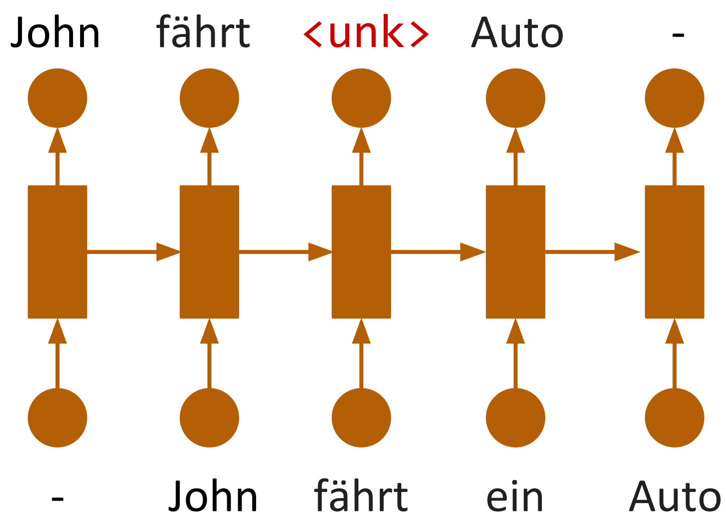
We employ a character level LSTM to then form an embedding for **“drives**”

Hybrid Units: Source side



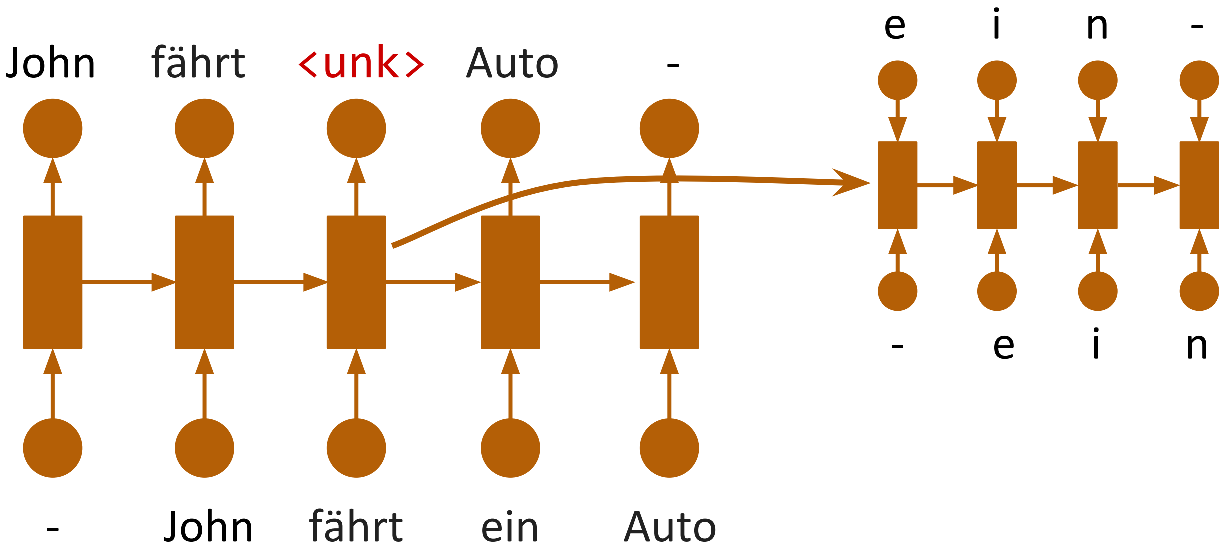
We employ a character level LSTM to then form an embedding for **“drives”**

Hybrid Units: Target side



If our word level model predicts an <unk>, we pass the associated hidden state to a character level LSTM!

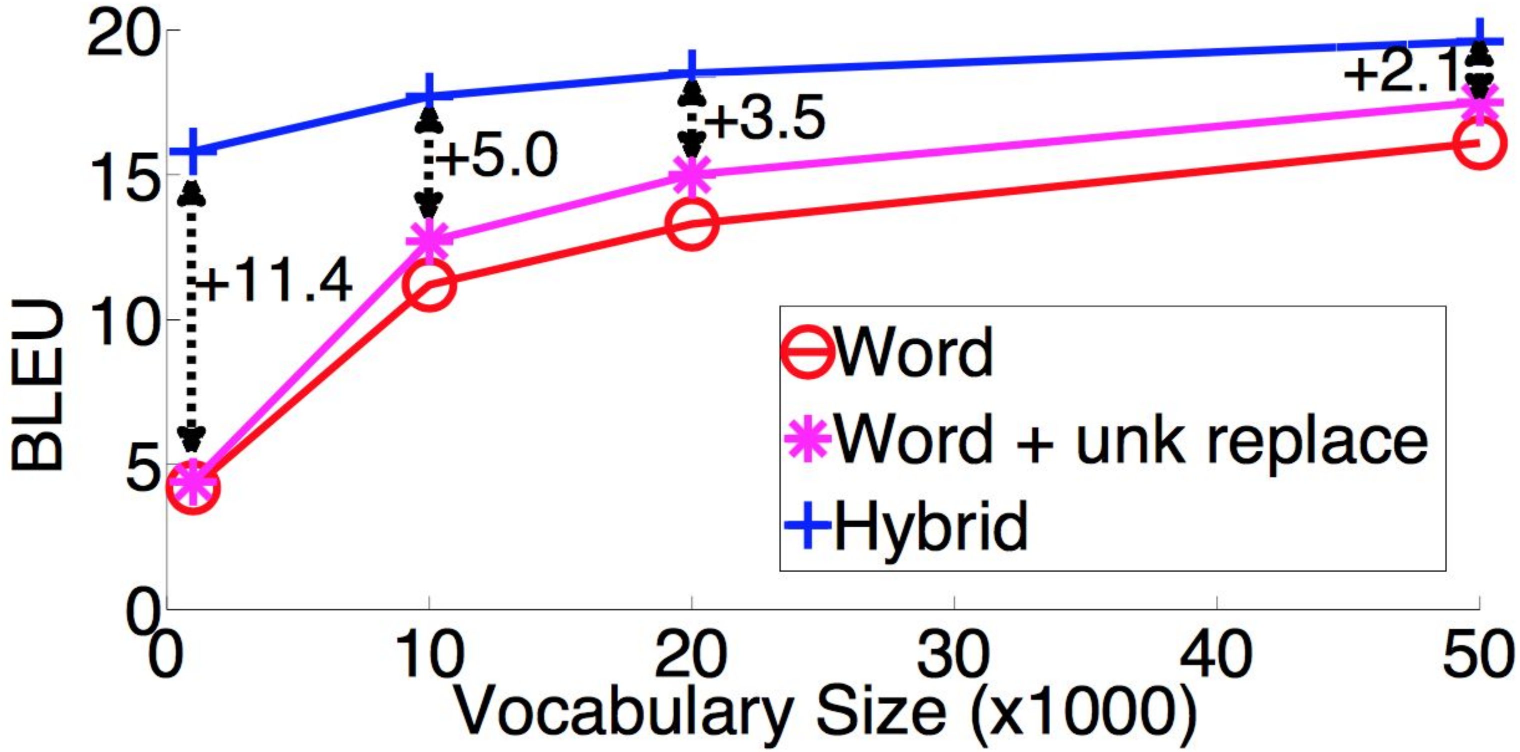
Hybrid Units: Target side



If our word level model predicts an <unk>, we pass the associated hidden state to a character level LSTM!

Hybrid Units

2.1 BLEU better!



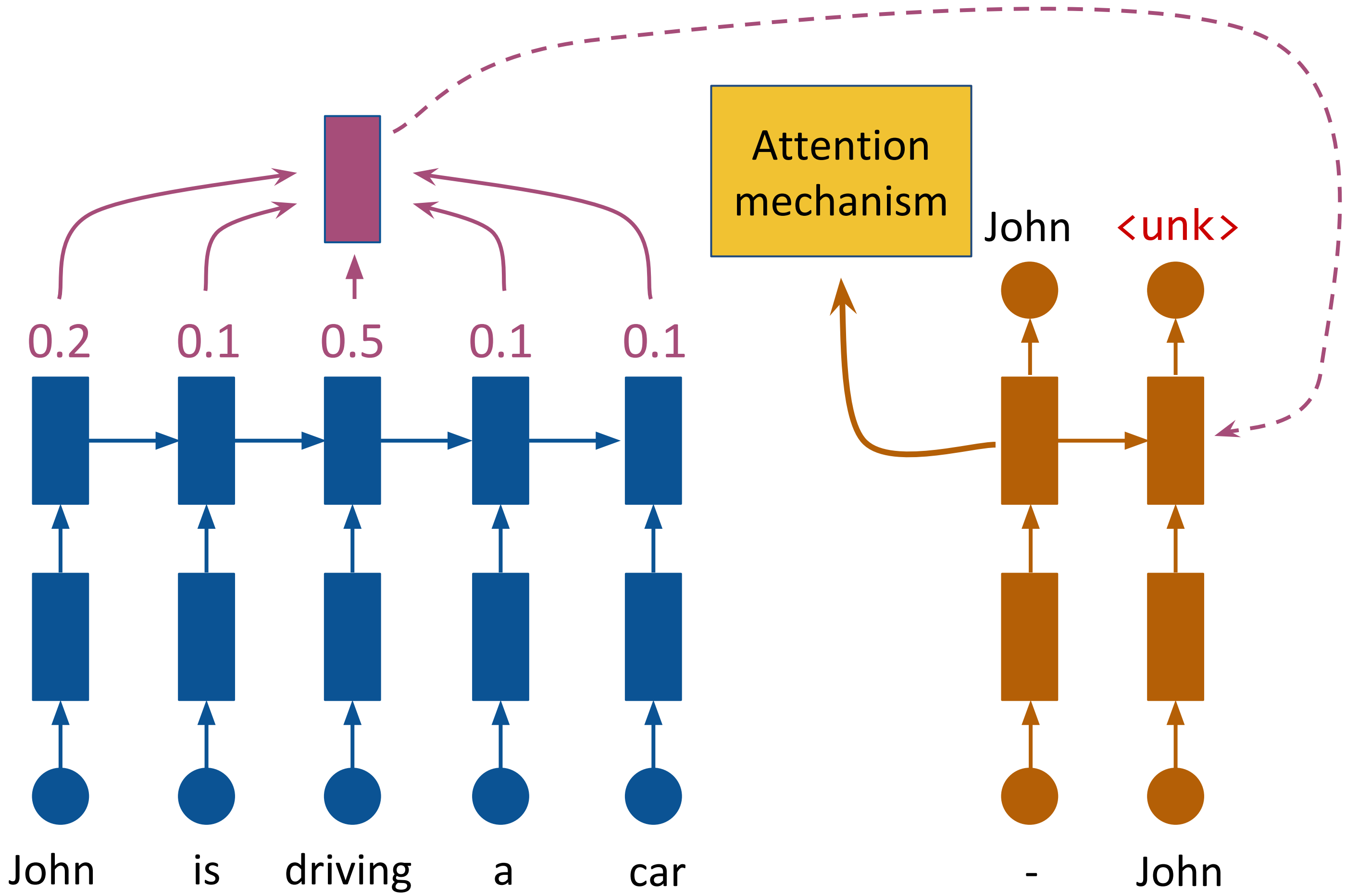
Summary: Subword Models

- BPE is by far the most commonly used method
- Character CNN and RNN representations do not perform well when applied on the target side
- A better solution is required that learns to produce morphologically correct segmentations while reducing the vocabulary size

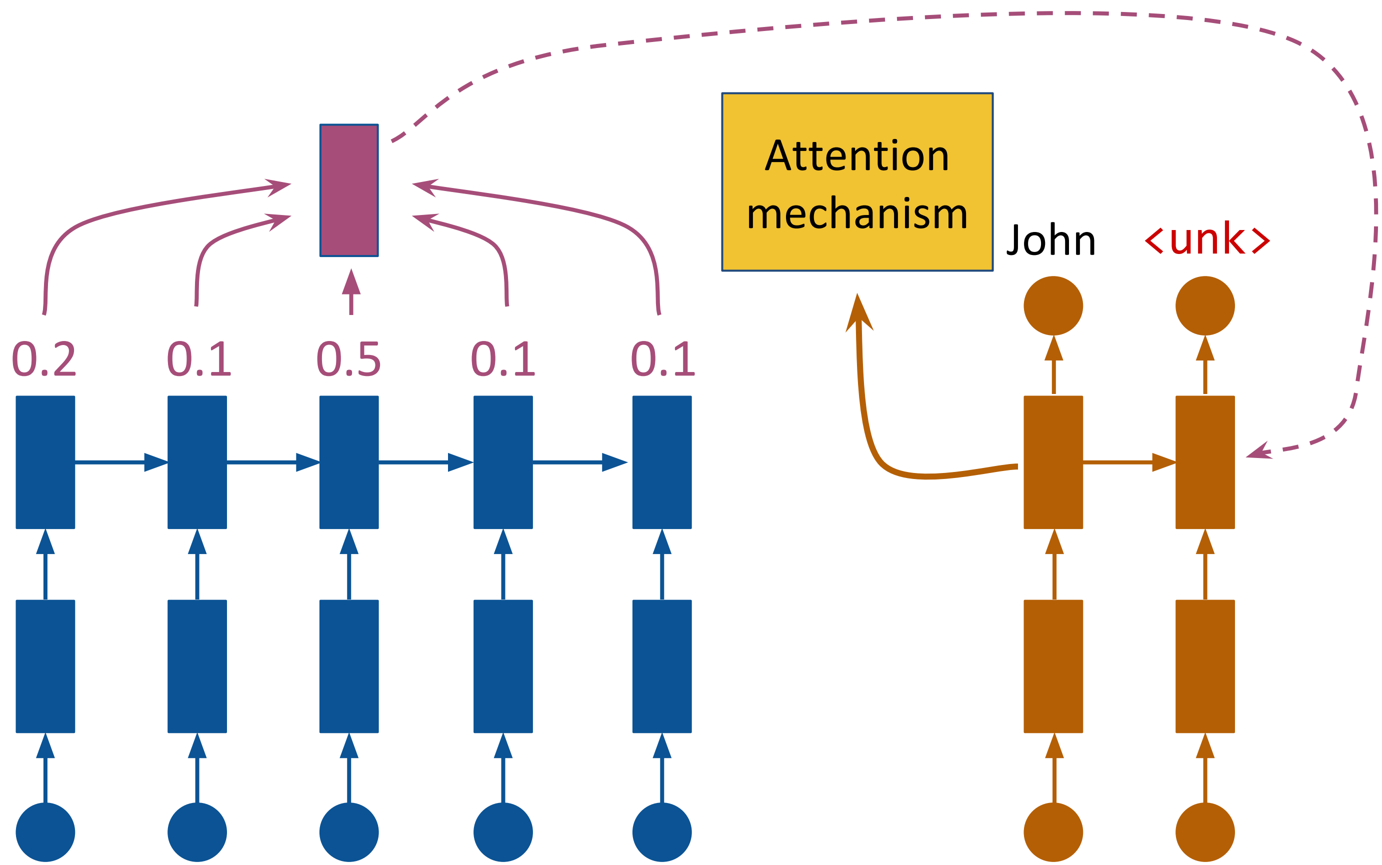
Practical Considerations

Dictionaries for unknown words

Unknown word dictionary

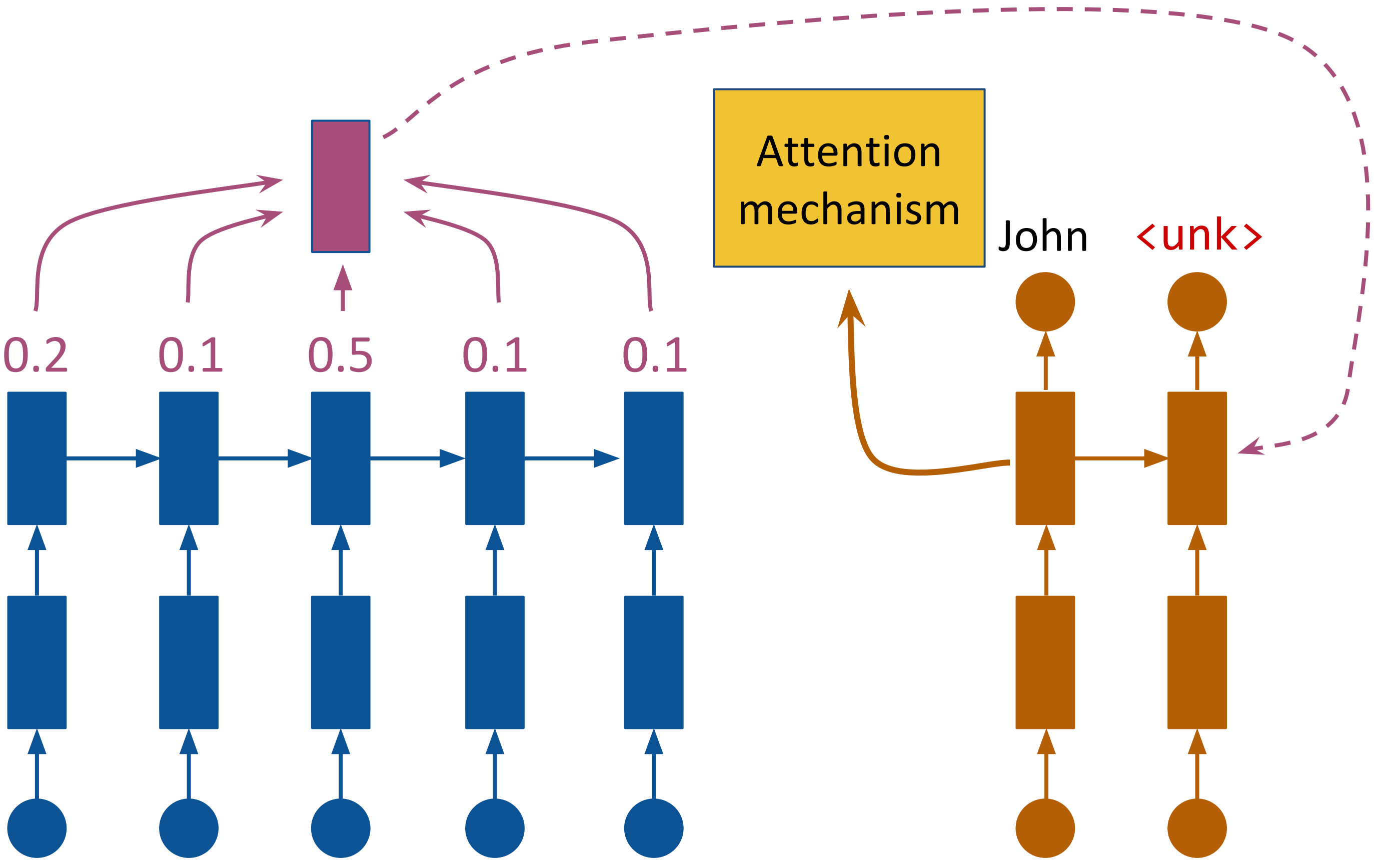


Unknown word dictionary



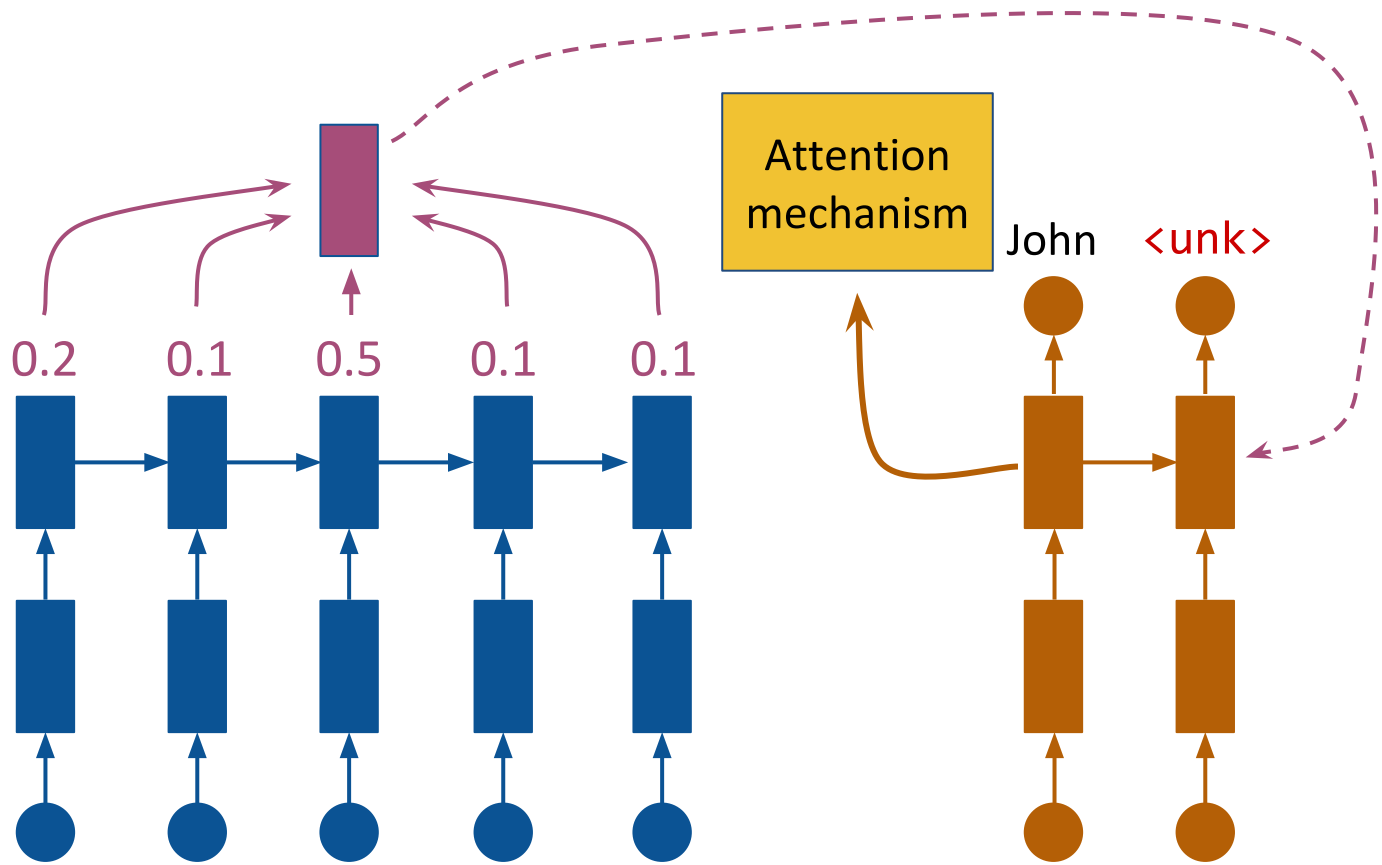
Even though the prediction is unknown, we have more information available!

Unknown word dictionary



Look at the english word with maximum attention!
In this case, it would be “driving”

Unknown word dictionary



Use an external dictionary to map
“driving” to “fährt”

Practical Considerations

Monolingual data

Monolingual Data

- Machine translation systems train on parallel corpora
- However, parallel data comes in limited amount compared to monolingual data

Monolingual Data

- Monolingual data is available in large quantity compared to parallel corpora between two languages
- Statistical MT makes use of large amount of monolingual data to train a language model

How can we use monolingual data in our NMT framework?

Using Monolingual Data

Several methodologies:

- Interpolation of translation score and LM score
- Multilingual MT
- Back translation

Practical Considerations

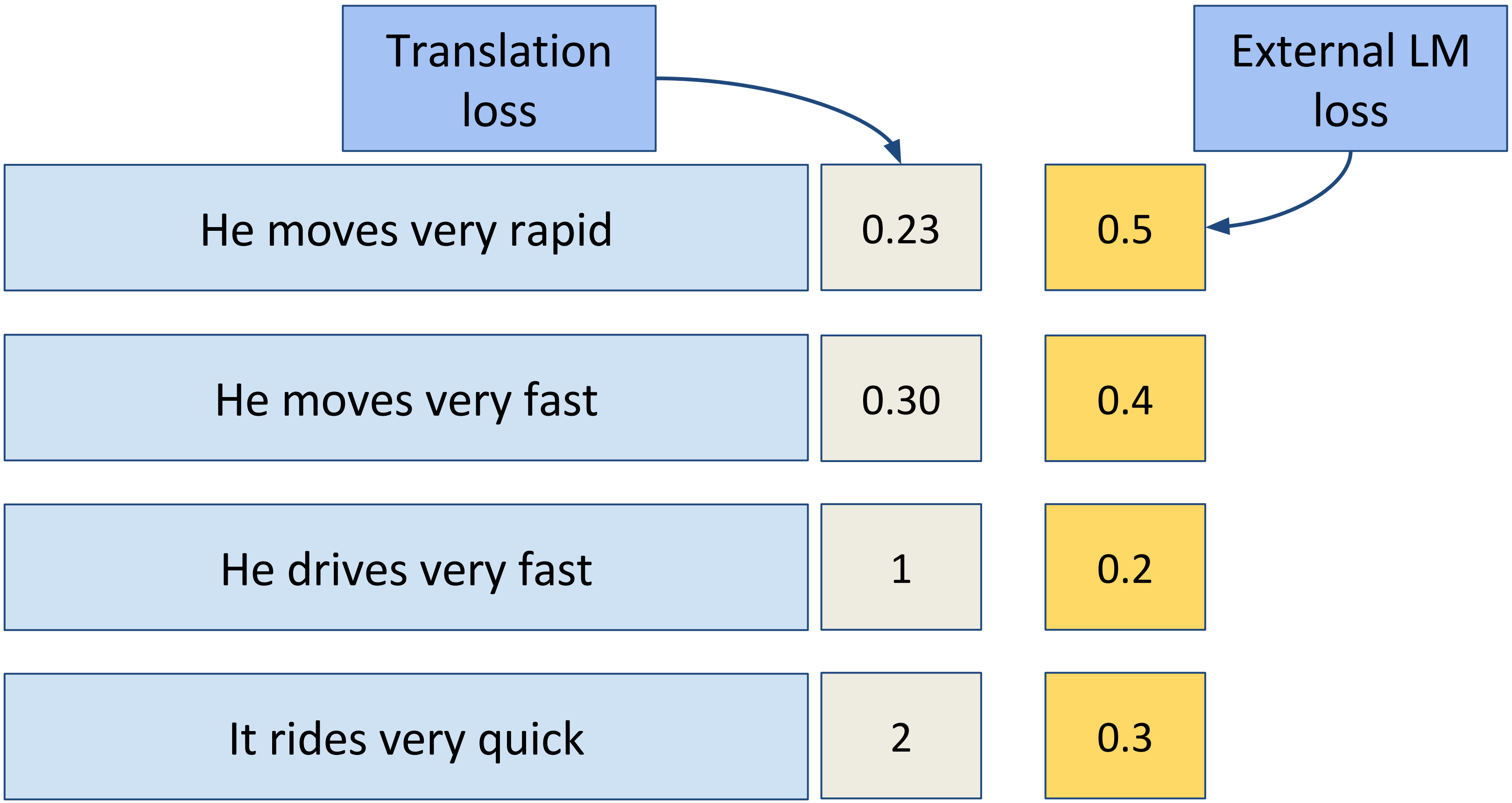
Monolingual data: Interpolation

Interpolation of Translation Scores and LM Scores

	Translation loss
He moves very rapid	0.23
He moves very fast	0.30
He drives very fast	1
It rides very quick	2

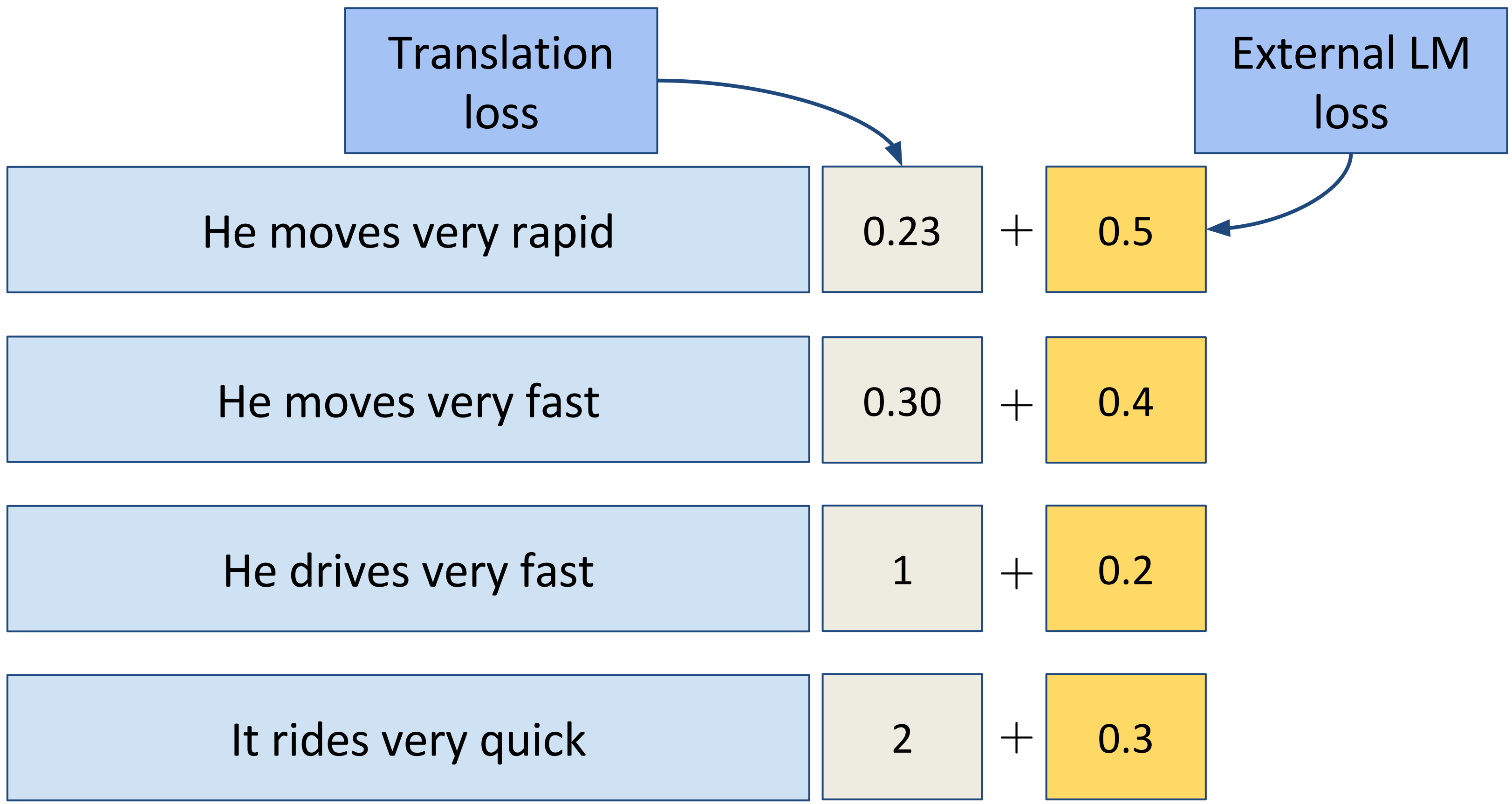
Consider we have four beams from our neural MT system

Interpolation of Translation Scores and LM Scores



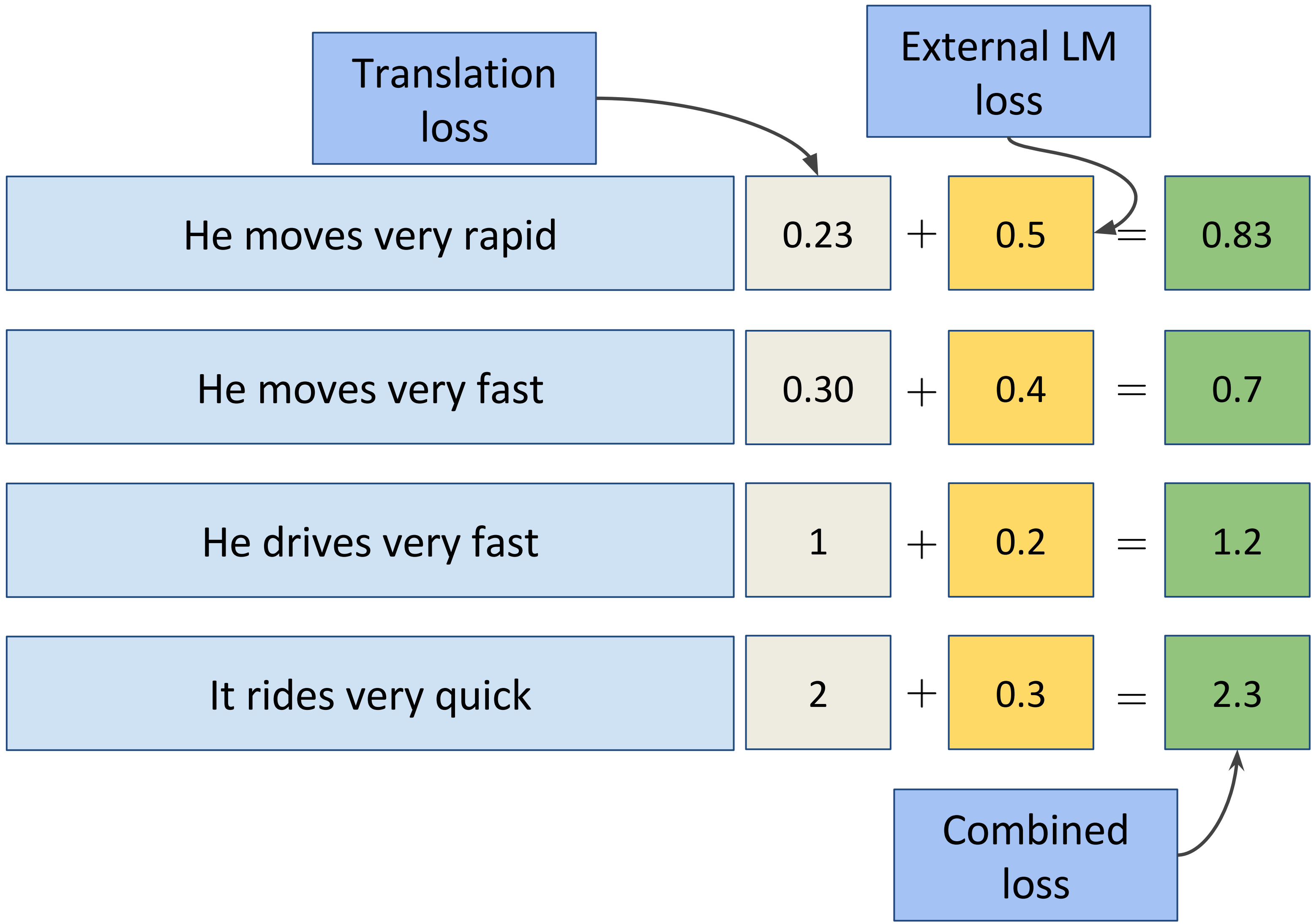
For all the translation options, we get scores from an **external** language model

Interpolation of Translation Scores and LM Scores



Combine translation and language models scores

Interpolation of Translation Scores and LM Scores



Interpolation of Translation Scores and LM Scores

Previous best from neural model

He moves very rapid	0.23	+	0.5	=	0.83
He moves very fast	0.30	+	0.4	=	0.7
He drives very fast	1	+	0.2	=	1.2
It rides very quick	2	+	0.3	=	2.3

Interpolation of Translation Scores and LM Scores

New best after re-ranking on combined scores

He moves very rapid	0.23	+	0.5	=	0.83
He moves very fast	0.30	+	0.4	=	0.7
He drives very fast	1	+	0.2	=	1.2
It rides very quick	2	+	0.3	=	2.3

Interpolation of Translation Scores and LM Scores

- Here, we combine translation model and language model scores with **equal** weights.
- Language model weight can be adjusted by tuning an *additional parameter* on a development set

[On Using Monolingual Corpora in Neural Machine Translation]

Interpolation of Translation Scores and LM Scores

- A better alternative in doing interpolation is to combine hidden states of decoder and LM while predicting the next word

- The hidden layer of the output takes as input the hidden state of the LM, hidden state of the previous target word, and the context vector

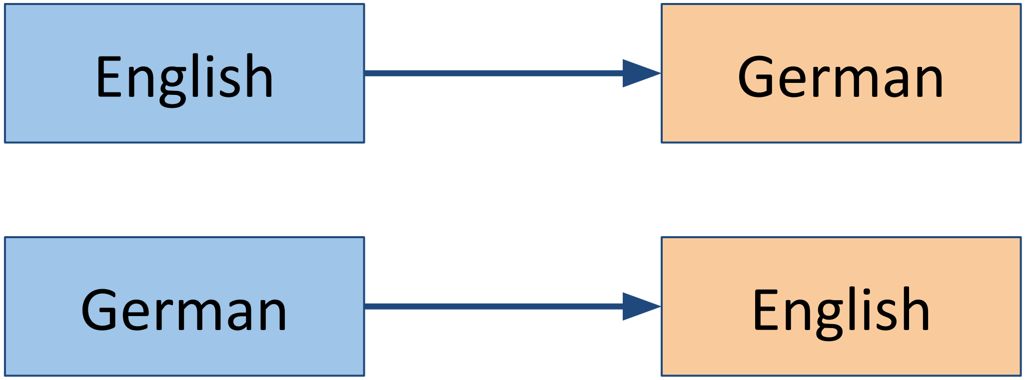
On Using Monolingual Corpora in Neural Machine Translation

Practical Considerations

Monolingual data: Multilingual systems

Multilingual systems

Generally, we train NMT systems from a single source language to a single target language



Multi-task Sequence to Sequence Learning

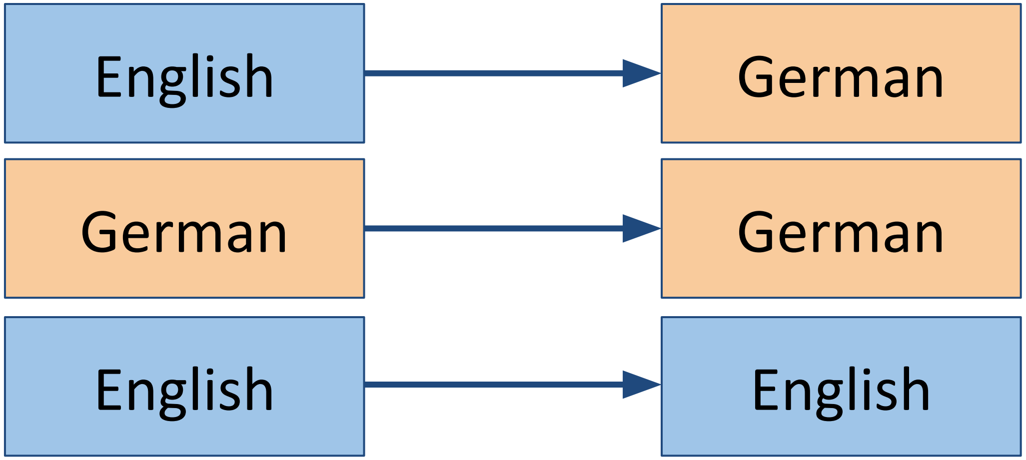
Multilingual systems

However, there is no inherent limitation in the architecture to have only one language for the encoder or the decoder!

[Multi-task Sequence to Sequence Learning]

Multilingual systems

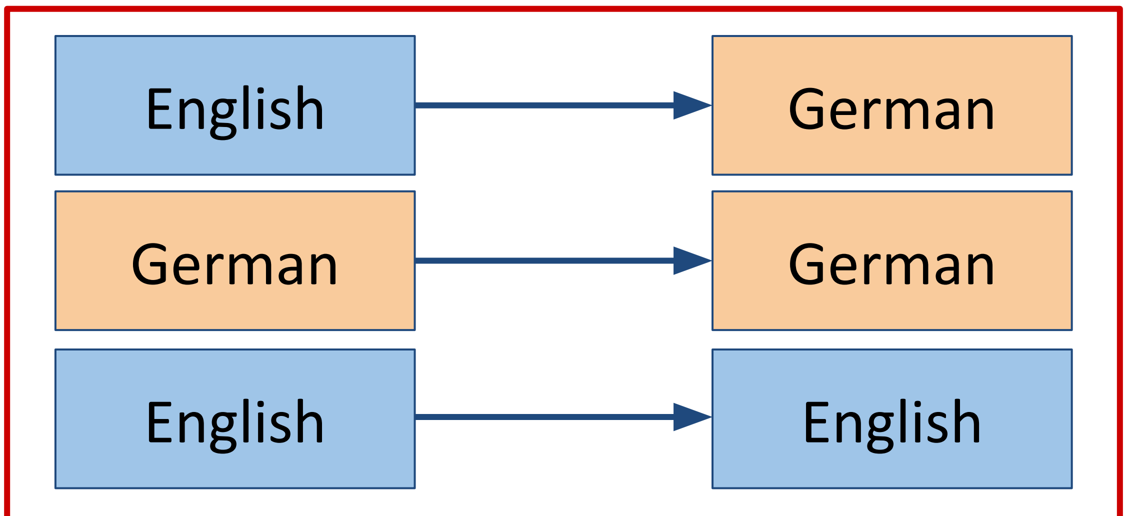
Idea: In addition to parallel data between English and German, add monolingual data from English to English and German to German



[Multi-task Sequence to Sequence Learning]

Multilingual systems

Idea: In addition to parallel data between English and German, add monolingual data from English to English and German to German

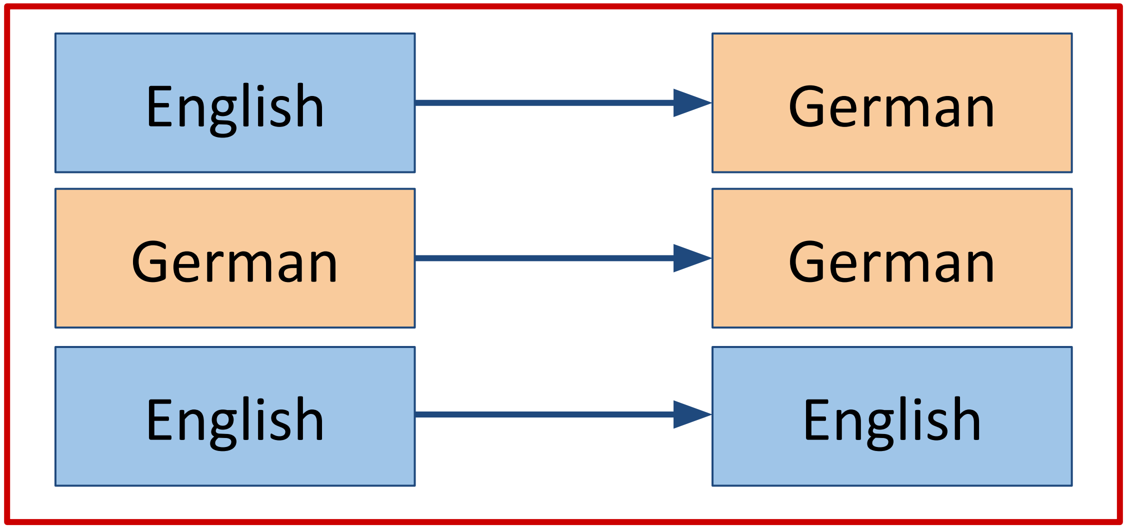


Single NMT system with mixed data

[Multi-task Sequence to Sequence Learning]

Multilingual systems

We are essentially learning three tasks together!
Each task helps the other two perform better



Single NMT system with mixed data

[Multi-task Sequence to Sequence Learning]()

Practical Considerations

Monolingual data: Synthetic data

Synthetic data

Consider a monolingual corpus for the target side - we have no source sentences unlike a parallel corpus!

[Improving Neural Machine Translation Models with Monolingual Data]()

Synthetic data

Consider a monolingual corpus for the target side - we have no source sentences unlike a parallel corpus!

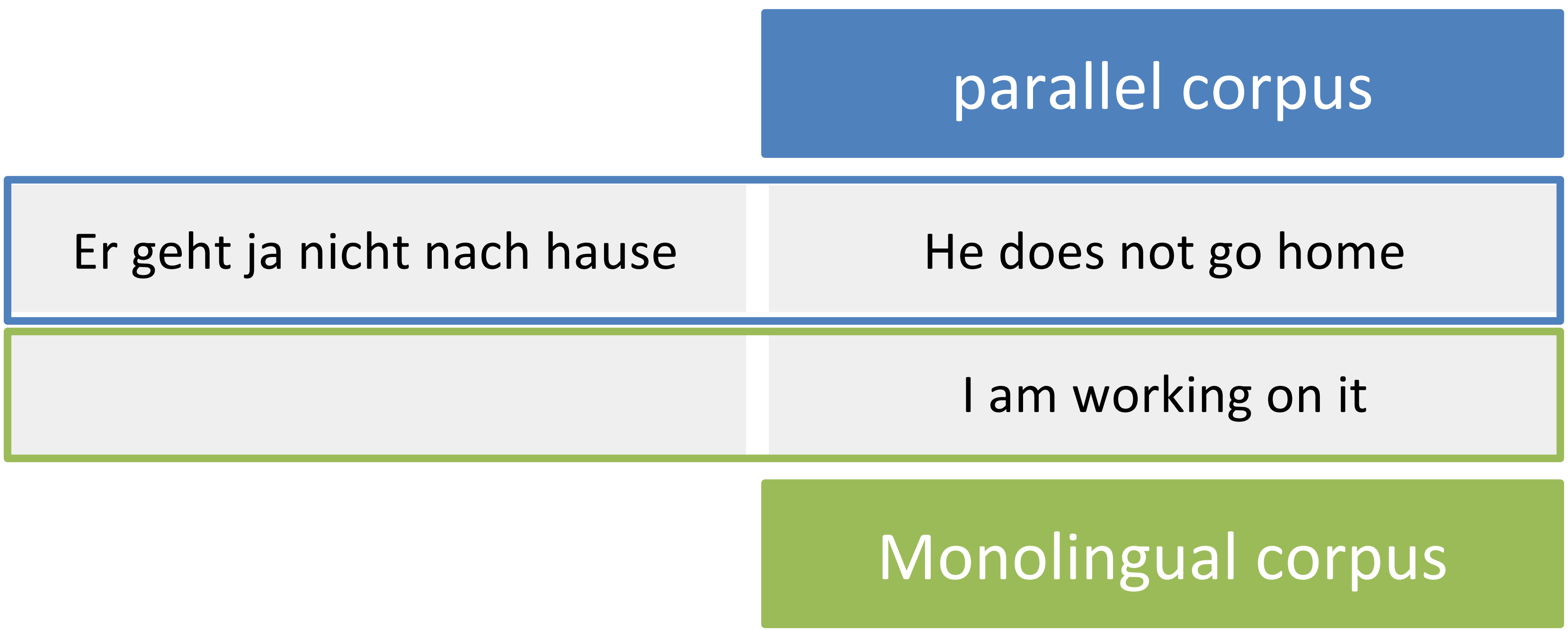
Two solutions to make monolingual data **look like** parallel data:

- add dummy source tokens
- produce synthetic source sentences using *back translation*

Improving Neural Machine Translation Models with Monolingual Data

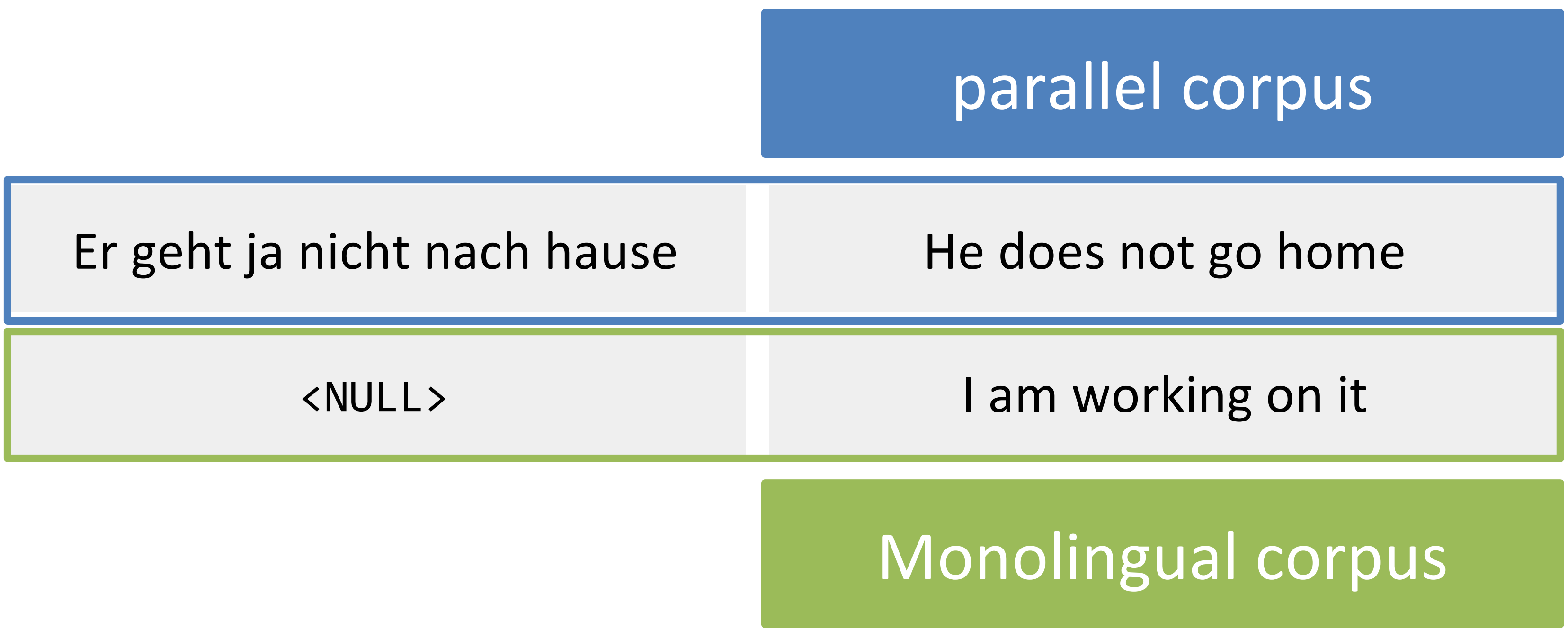
Monolingual Data: Dummy Tokens

Add dummy token on source side and **mix** the data with the parallel corpus



Monolingual Data: Dummy Tokens

Add dummy token on source side and **mix** the data with the parallel corpus



Monolingual Data: Dummy Tokens

Add dummy token on source side and **mix** the data with the parallel corpus

parallel corpus

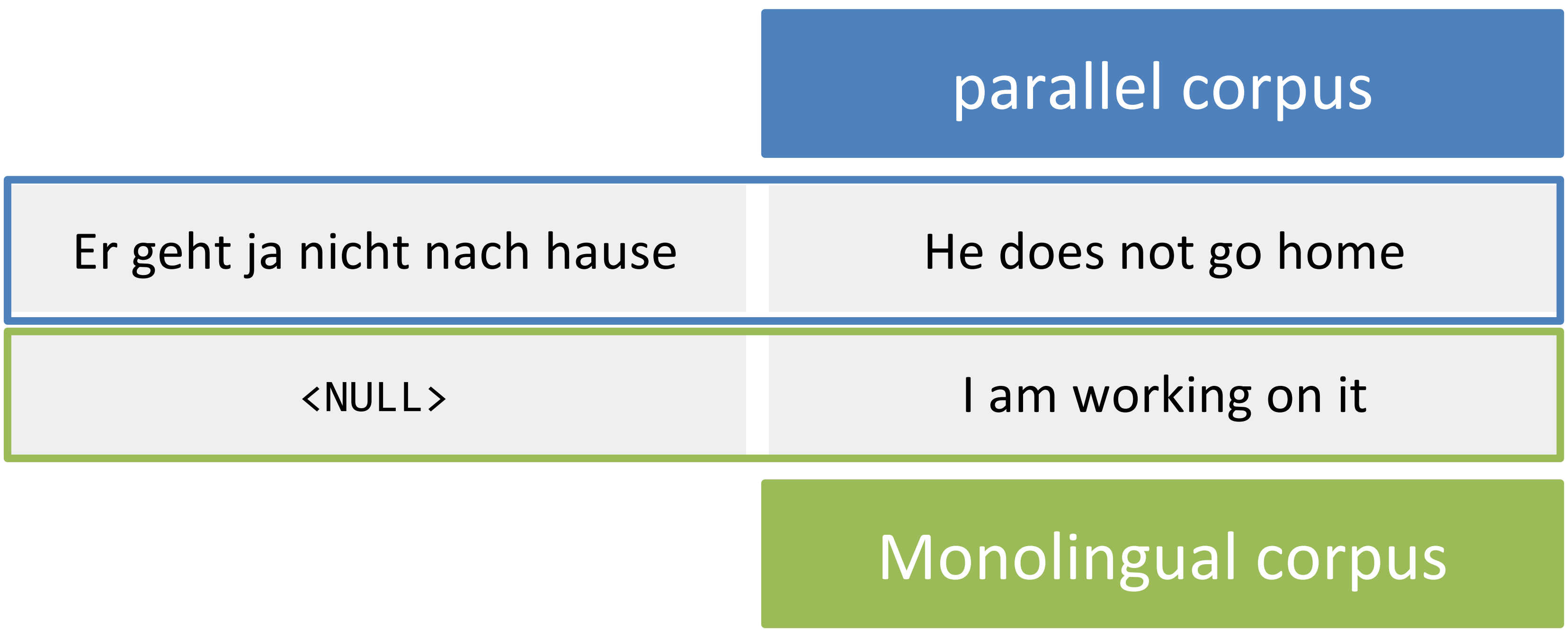
Er geht ja nicht nach hause	He does not go home
<NULL>	I am working on it

Monolingual corpus

Freeze encoder/attention layer while processing dummy parallel corpus

Monolingual Data: Dummy Tokens

Add dummy token on source side and **mix** the data with the parallel corpus



Improves translation quality up to 1.0 BLEU points

Monolingual Data: Synthetic Data

Synthetic data: back translate monolingual data into source language using a **target to source machine translation** system

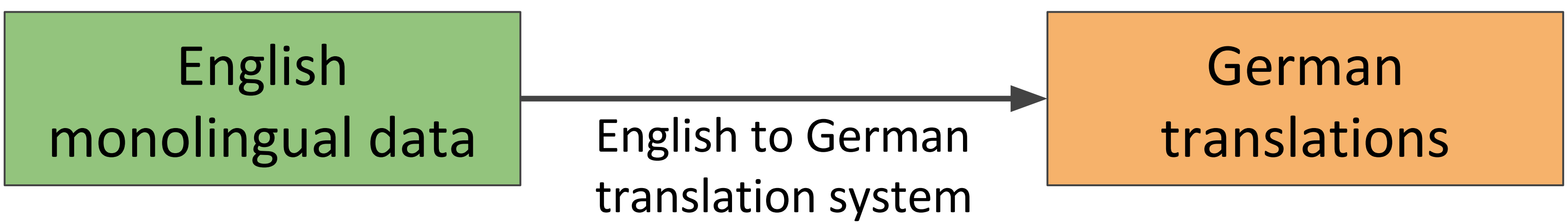
Monolingual Data: Synthetic Data

Synthetic data: back translate monolingual data into source language using a **target to source machine translation** system

English monolingual data

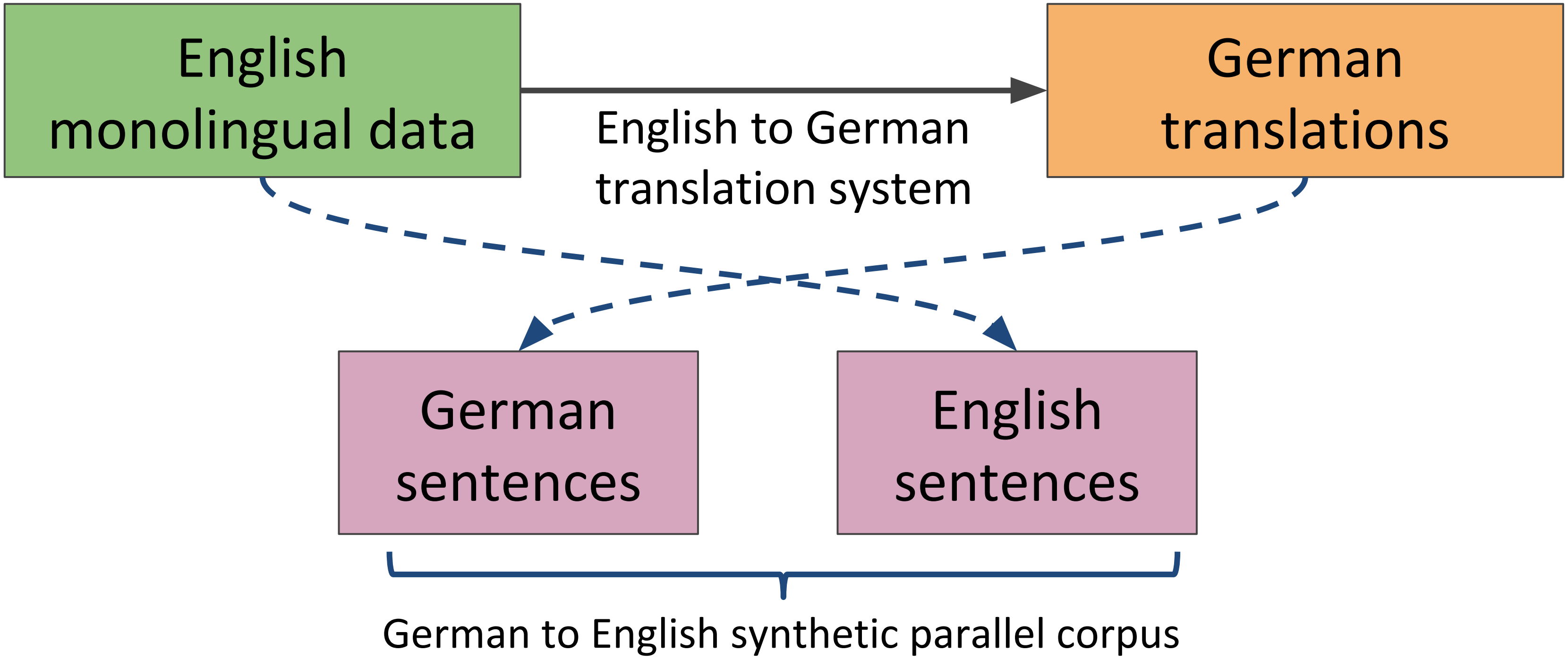
Monolingual Data: Synthetic Data

Synthetic data: back translate monolingual data into source language using a **target to source machine translation** system

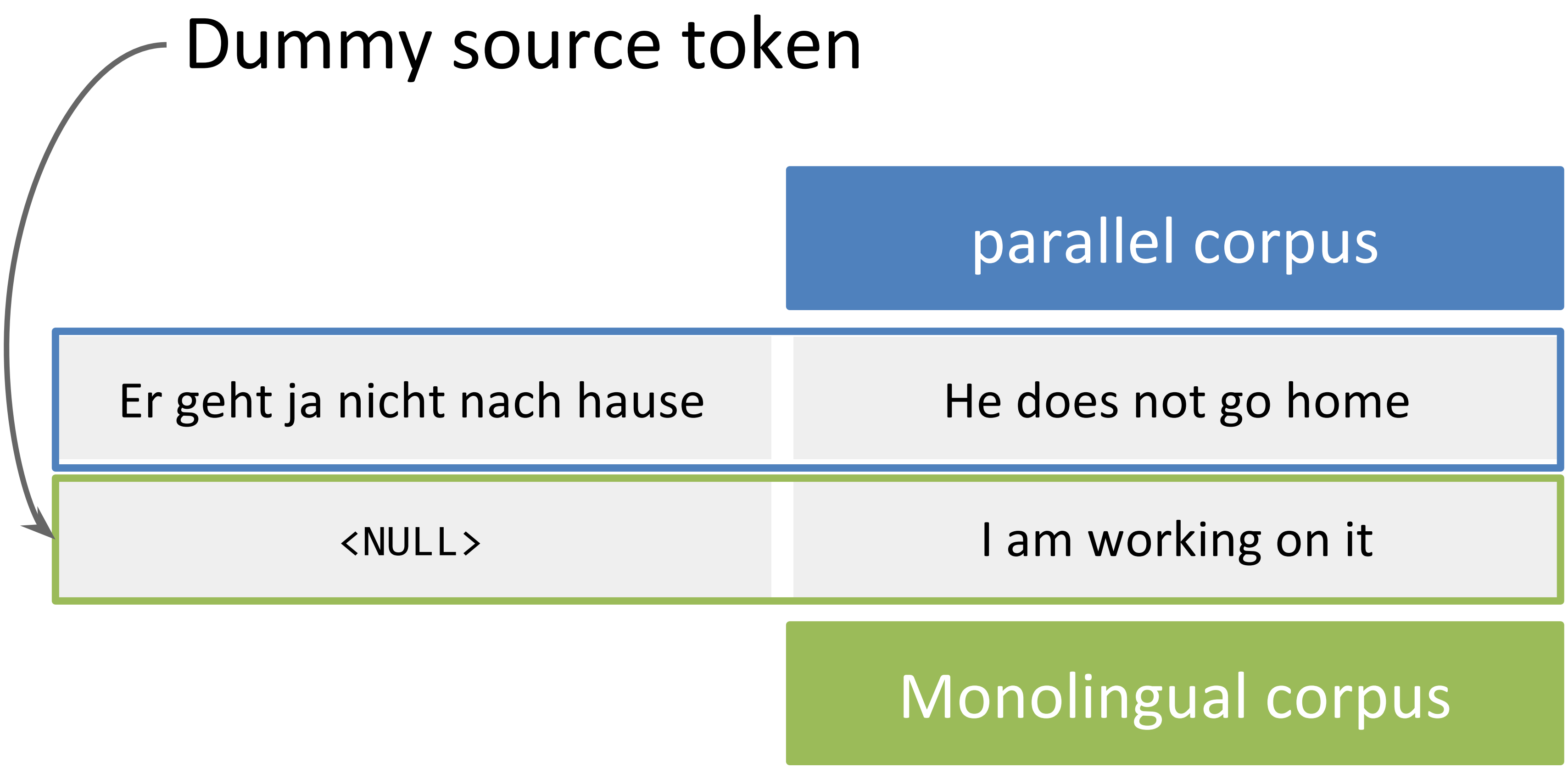


Monolingual Data: Synthetic Data

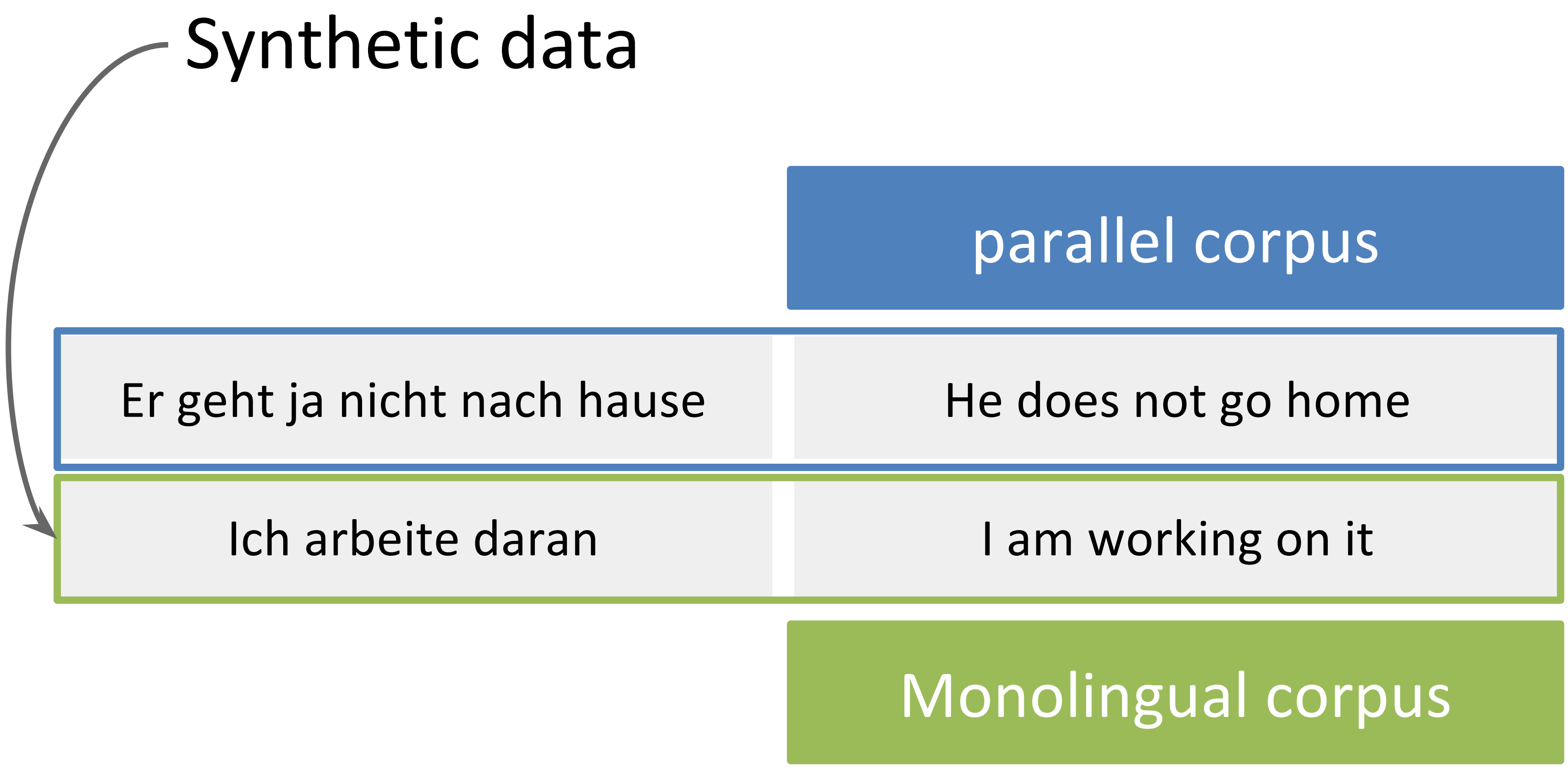
Synthetic data: back translate monolingual data into source language using a **target to source machine translation** system



Monolingual Data: Synthetic Data



Monolingual Data: Synthetic Data



Monolingual Data: Synthetic Data

- Train the NMT system on a combination of synthetic and parallel corpora
- Large performance improvements from **2.1-3.4** BLEU points

Summary: Monolingual Data

- Effective use of monolingual data gives a performance improvements of 2-3 BLEU points
- It has become absolutely necessary in competitions, such as WMT

Practical Considerations

Data setup

Data setup

- Training set
  - a set of ***parallel*** **sentences** to train our model on
- Tune/dev/held-out set
  - model tends to overfit
  - use a **small set of *parallel* sentences** to test the model during training
- Test set
  - a **small set of *source* sentences** to test the final model

Data setup

- Training set
  - a set of ***parallel* sentences** to tr
- Tune/dev/held-out set
  - model tends to overfit
  - use a **small set of *parallel* sente** model during training
- Test set
  - a **small set of *source* sentences** to test the final model

It is important that each of these sets have the same “distribution” i.e. be from the same domain

Practical Considerations

Best Practices: Initialization

Initialization

So far, we have initialized our weight matrices with (small) random numbers

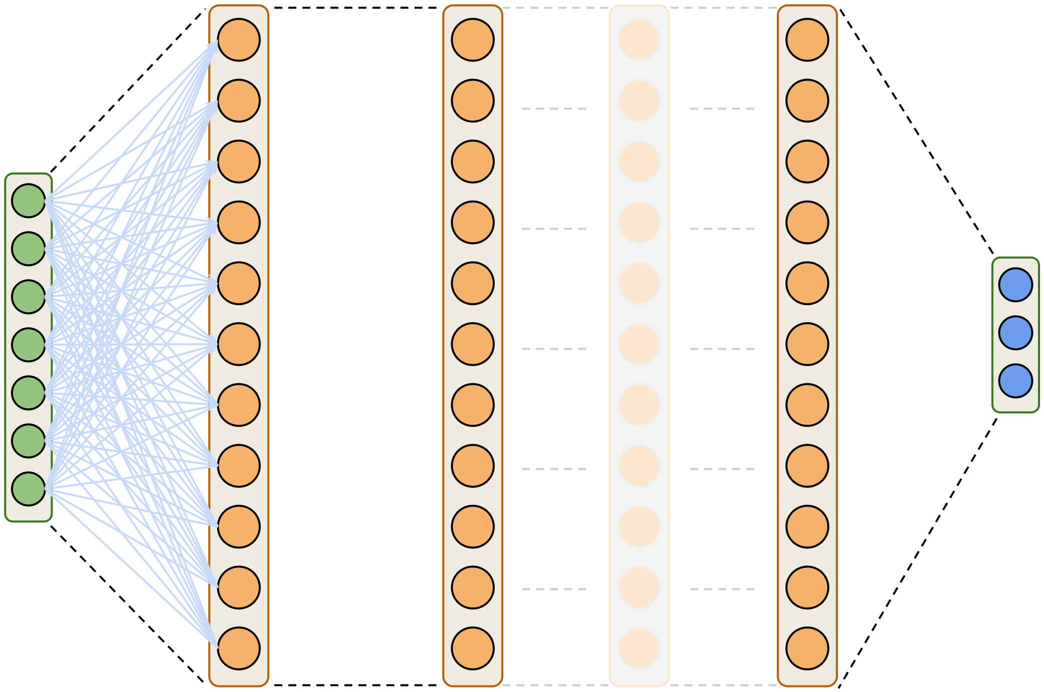
Initialization

So far, we have initialized our weight matrices with (small) random numbers

Q: Why should we never use zero initialization with neural networks?

Initialization

Q: Why should we never use zero initialization with neural networks?



Initialization

Q: Why should we never use zero initialization with neural networks?

A: All the neurons see the same input - and with the same weight matrices, they will make the exact same decisions!

Initialization

There is a better way to initialize weight matrices other than random numbers: **Xavier initialization**

Initialization

There is a better way to initialize weight matrices other than random numbers: **Xavier initialization**

Different for each layer - depends on the number of connections coming in and going out!

Initialization

Usually when we pick random numbers, we pick them from a uniform distribution

Initialization

Usually when we pick random numbers, we pick them from a uniform distribution

Xavier initialization says we should pick the random numbers from a distribution with zero mean and variance:

$$Var(W) = \frac{2}{n_{in} + n_{out}}$$

Initialization

Works very well in practice - was actually an enabler in training deeper networks at some point in time!

Practical Considerations

Best Practices: Batch size

Batch Size

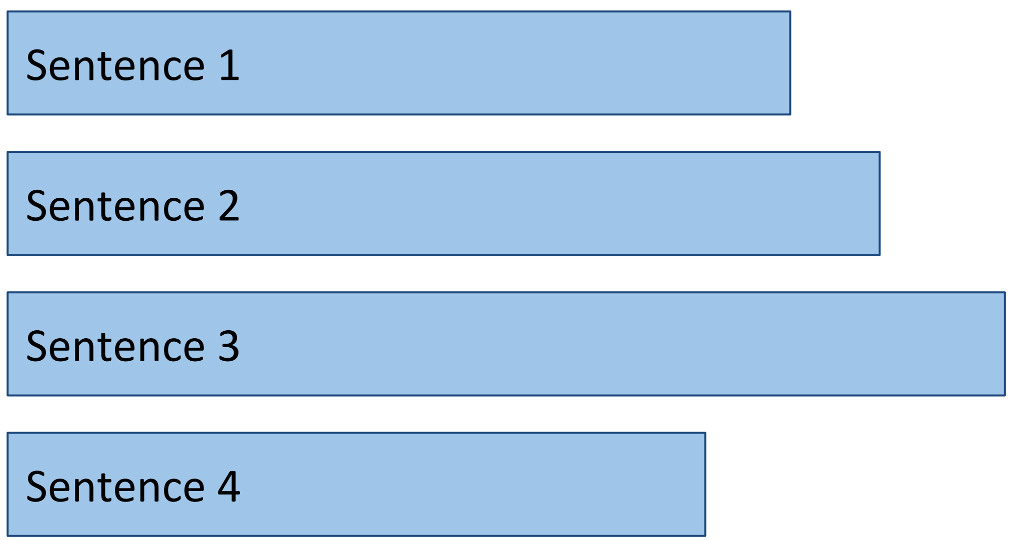
- In standard practice, a minibatch size of 80 sentences is used
  - shuffle training data
  - divide it in batches of 80 sentences
  - run training and update parameters for every batch
- Bigger batches result in more stable updates but slower the training process

Practical Considerations

Best Practices: Padding

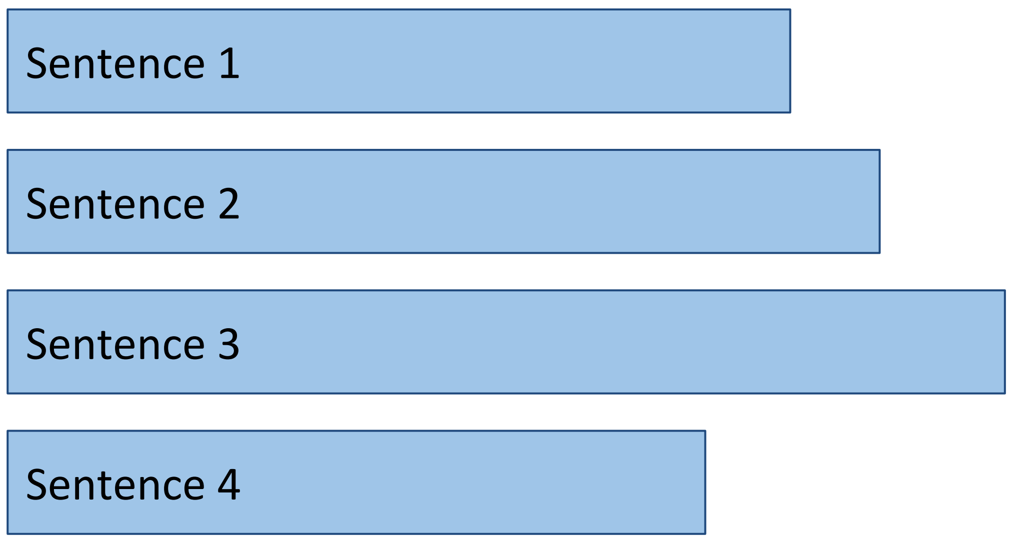
Padding

- Input sentences are of varied length
- Need a fixed length to define a fixed size of weight matrices



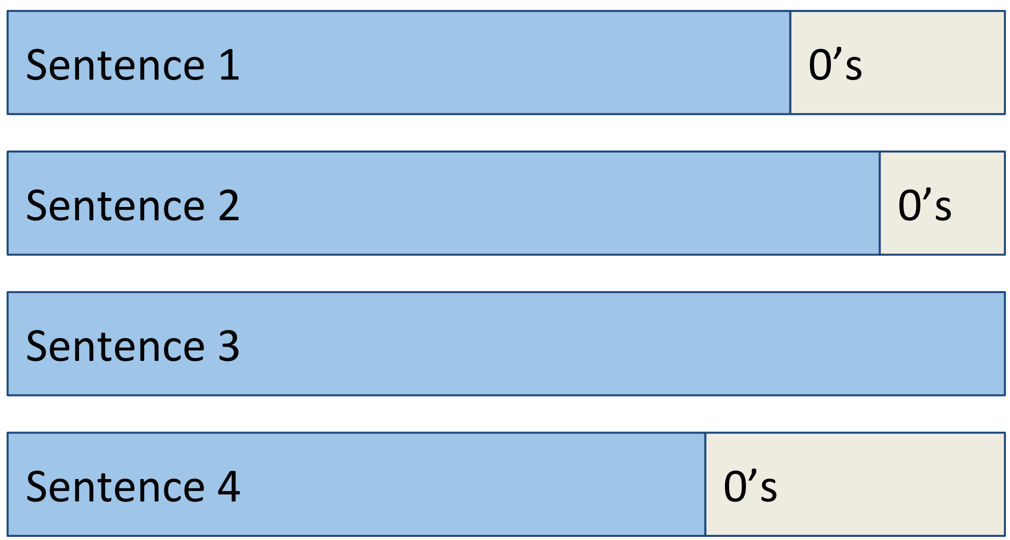
Padding

Solution: Pad smaller sentences with 0's



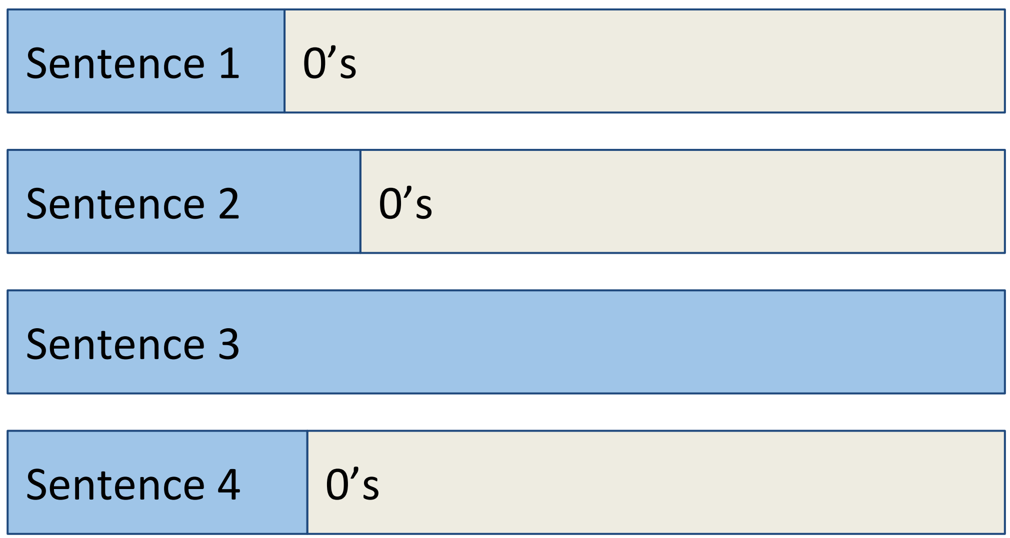
Padding

Solution: Pad smaller sentences with 0’s



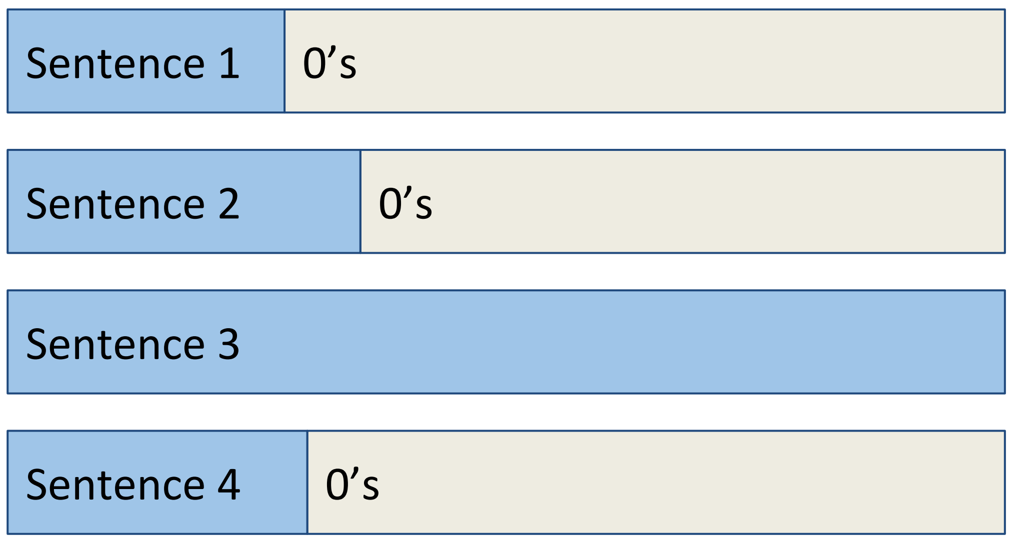
Padding

Problem: What if one sentence is very long in a batch?



Padding

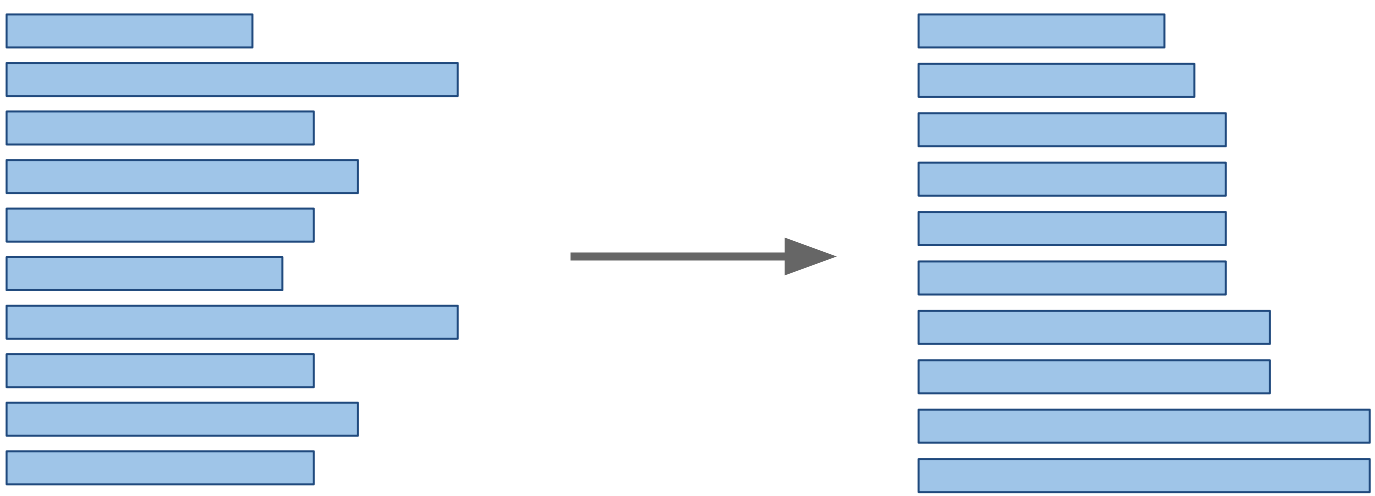
Problem: What if one sentence is very long in a batch?



A lot of wasted computation!

Padding

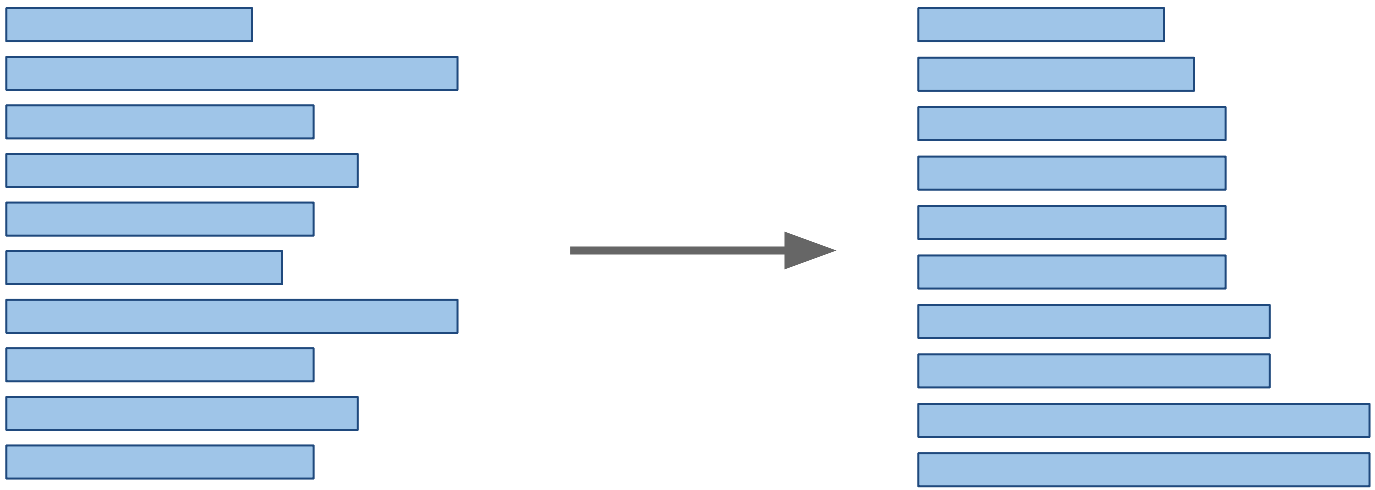
- Alternatively, sort all sentences by length
- Create minibatches by putting sentences of similar length together



- Limit wasted computation in every minibatch

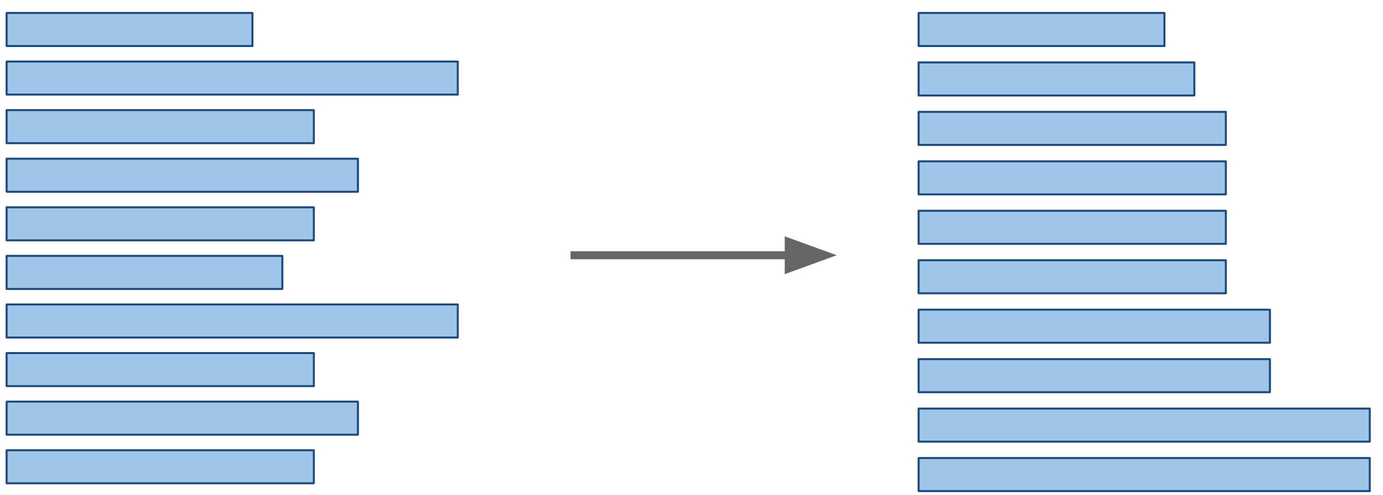
Padding

Q: Any problems with this solution?



Padding

Q: Any problems with this solution?



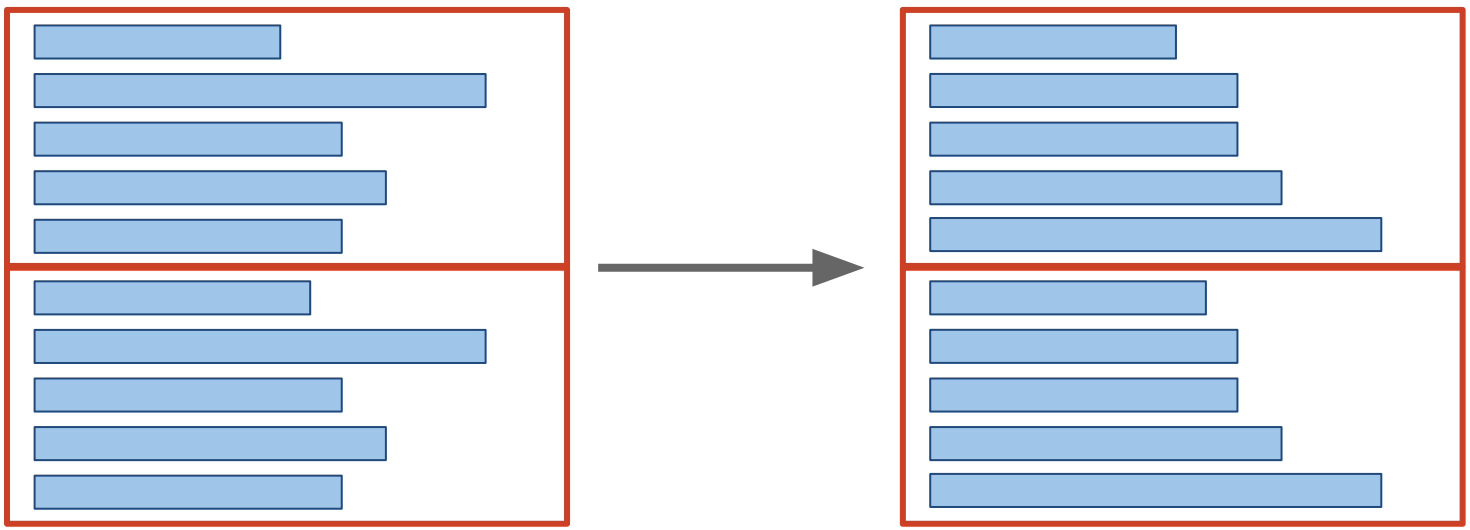
A: Yes! We are inducing a bias so that the model sees all short sentences early and all long sentences later

Padding

Solution: Sort sentences in a **maxi-batch**

Padding

Solution: Sort sentences in a **maxi-batch**



Now choose minibatches from each maxibatch

Practical Considerations

Toolkits

Toolkits

- [Nematus]
  - Python based, Theano
  - used in best competition grade system (WMT17)
- [OpenNMT]
  - Lua based, Torch, PyTorch
  - flexible to incorporate new features
  - Image to text is available
- Many more other toolkits are available

OpenNMT

Preprocess the data:

```
th preprocess.lua -train_src
 data/src-train.txt -train_tgt
 data/tgt-train.txt -valid_src
 data/src-val.txt -valid_tgt
 data/tgt-val.txt -save_data data/demo
```

- Train and validation data of source and target languages consist of one line per sentence with words separated by space
- Validation data size is usually 2000-5000 sentences

OpenNMT

Preprocess the data:

```
th preprocess.lua -train_src
 data/src-train.txt -train_tgt
 data/tgt-train.txt -valid_src
 data/src-val.txt -valid_tgt
 data/tgt-val.txt -save_data data/demo
```

- Note: you may need to tokenize (punctuation separator) the sets before preprocessing it

OpenNMT

Train a model:

```
th train.lua -data data/demo-train.t7
-save_model demo-model
```

- Takes the **t7** file generated using the preprocess script to train the model
- Since we have not specified any other parameters, default model will consist of 2-LSTM layers with 500 hidden units

OpenNMT

Translate:

```
th translate.lua -model
demo-model_epochX_PPL.t7 -src
data/src-test.txt -output pred.txt
```

- Once model is trained, input the test data to get translations

Summary

- Bidirectional LSTMs provide summary of source sentence in both left to right and right to left direction
- Dropout is an easy and effective way to avoid overfitting of the model
- Residual connections avoid the issue of information decay in deep models
- Ensemble enables the use of multiple models to improve translation performance at test time

Summary

- Subword-based models particularly BPE is practically useful to limit vocabulary and to handle unknown words during testing
- Using synthetic data to benefit from a large monolingual target data significantly improves the translation performance