Natural Language Processing with Deep Learning

CS224N/Ling284

Christopher Manning

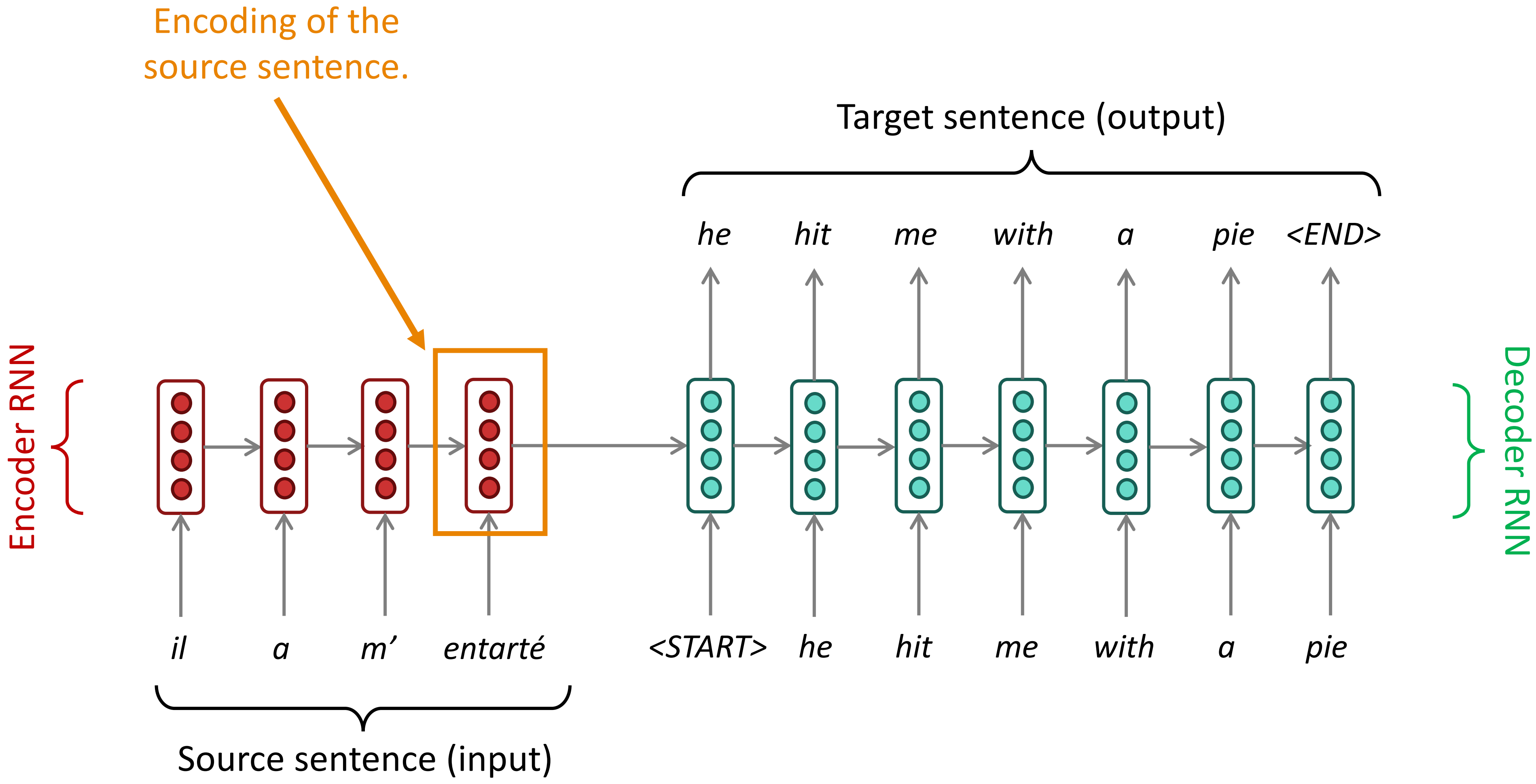
Lecture 8: Final Projects; Practical Tips

Lecture Plan

Lecture 8: Finish last time – final Projects – practical tips!

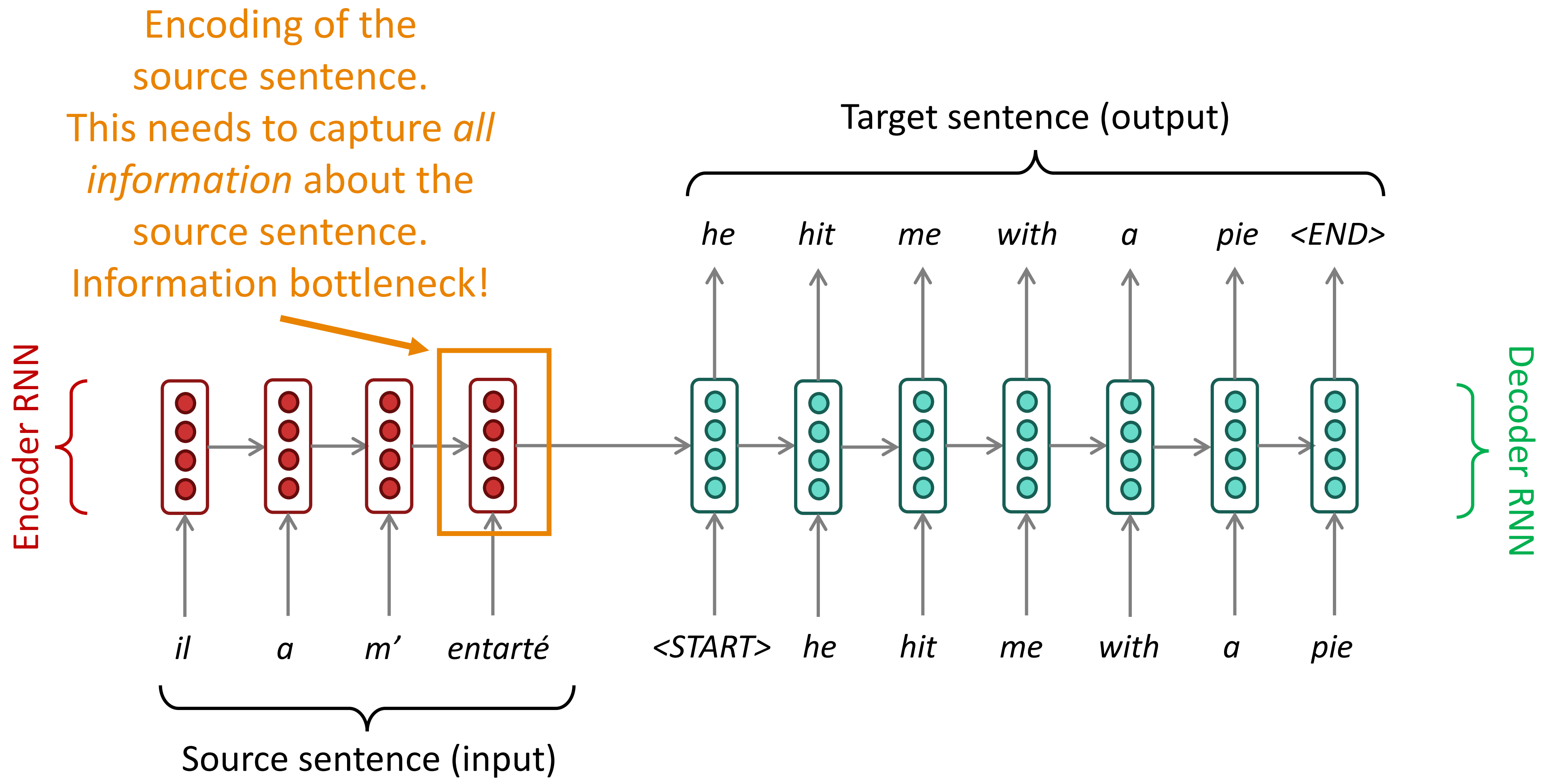
1. Attention [25 mins]
2. Final bit of neural machine translation [10 mins]
   – Mini Break –
3. Final project types and details; assessment revisited [15 mins]
4. Finding research topics; a couple of examples [20 mins]
5. Finding data [10 mins]
6. Care with datasets and in model development [10 mins]

1. Why attention? Sequence-to-sequence: the bottleneck problem



Problems with this architecture?

1. Why attention? Sequence-to-sequence: the bottleneck problem



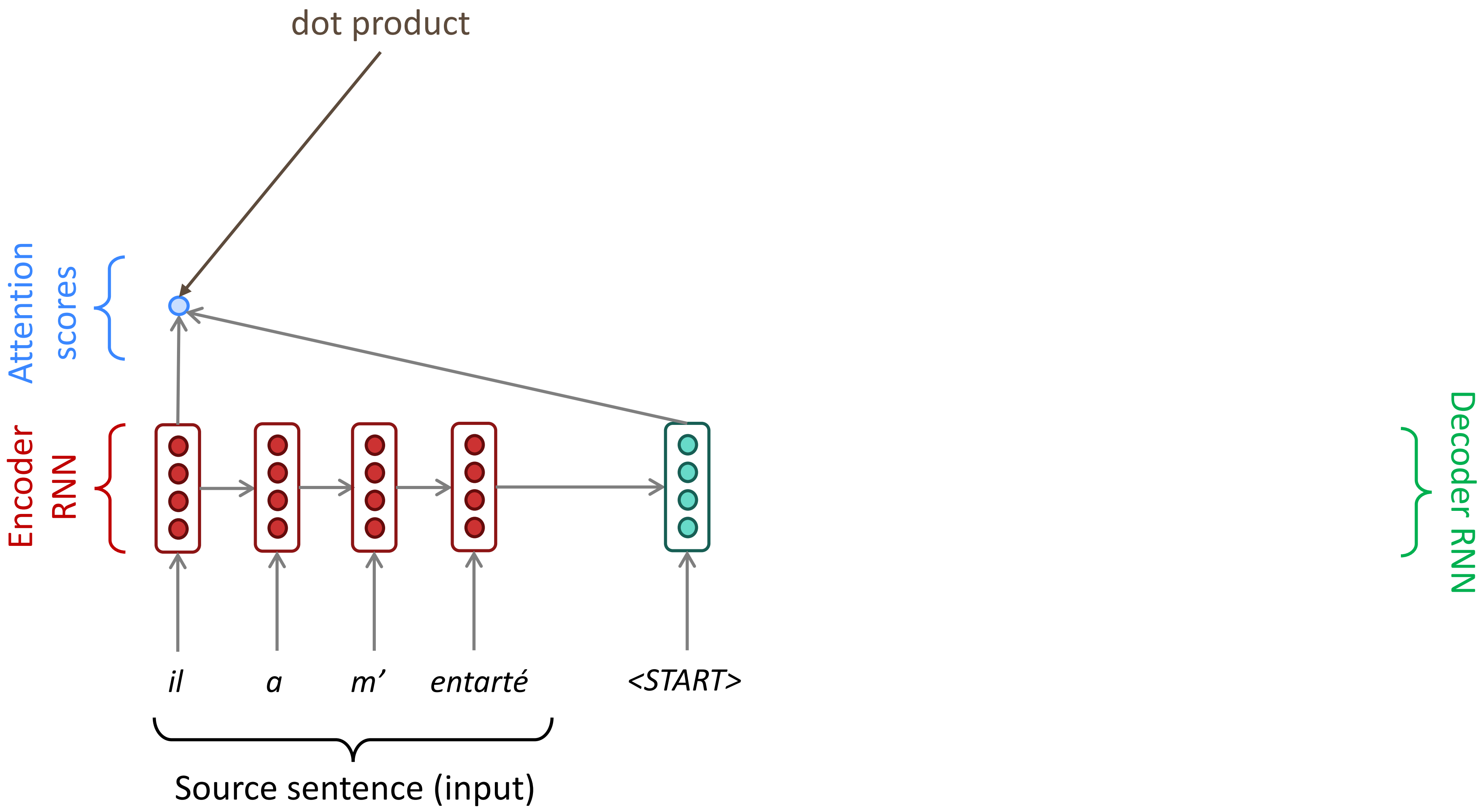
Attention

- **Attention** provides a solution to the bottleneck problem.

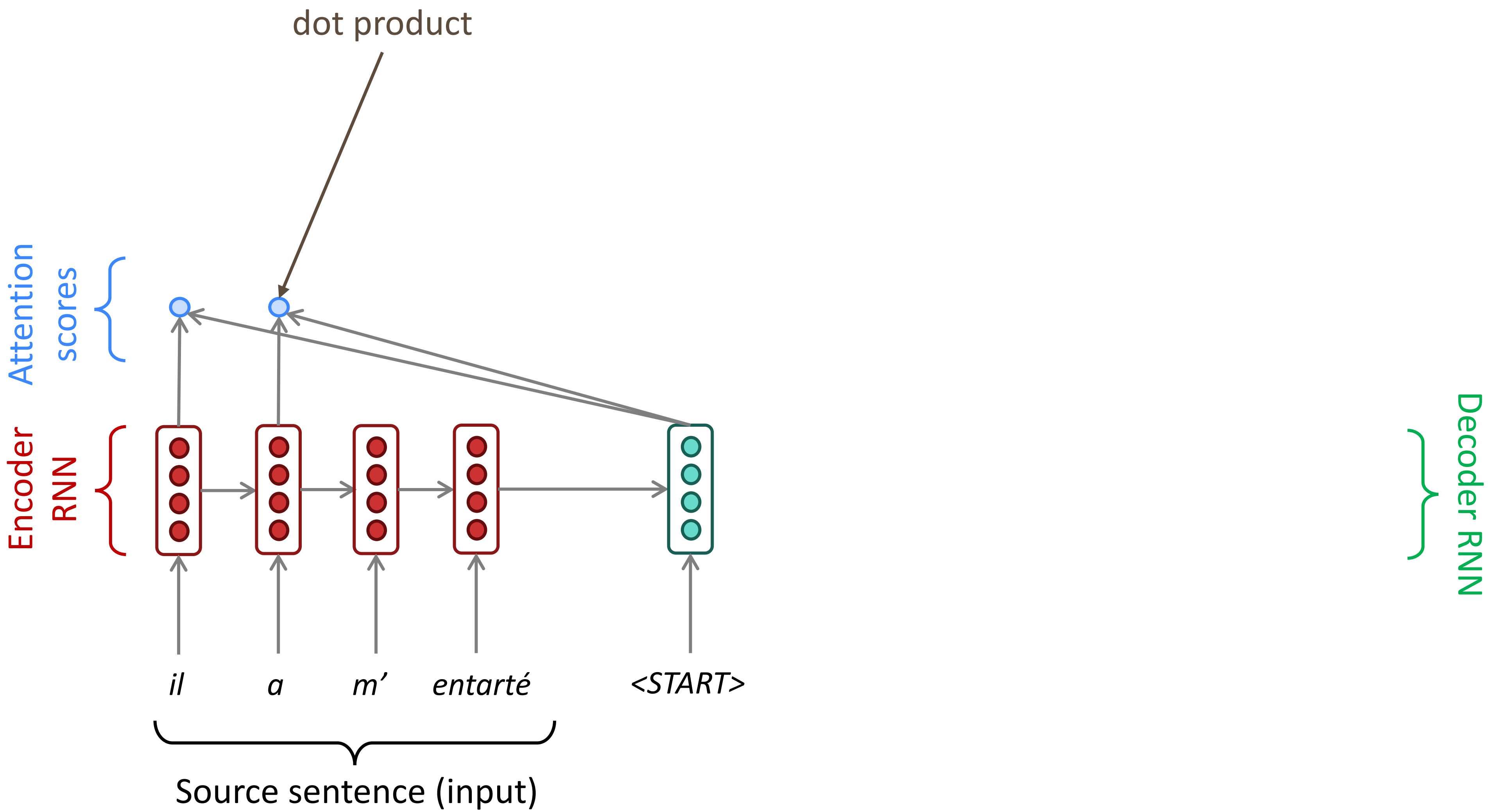
- Core idea: on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence

- First, we will show via diagram (no equations), then we will show with equations

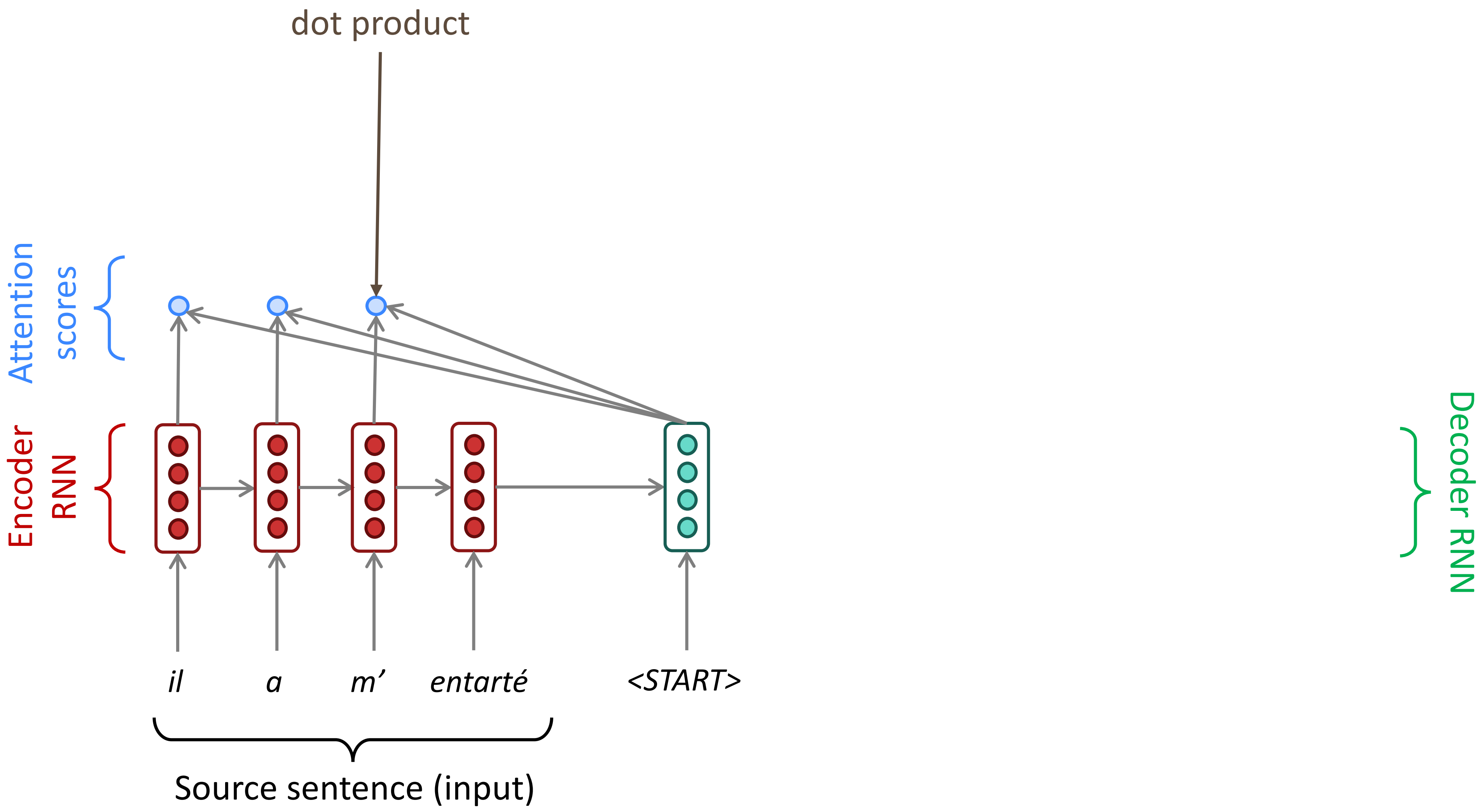
Sequence-to-sequence with attention



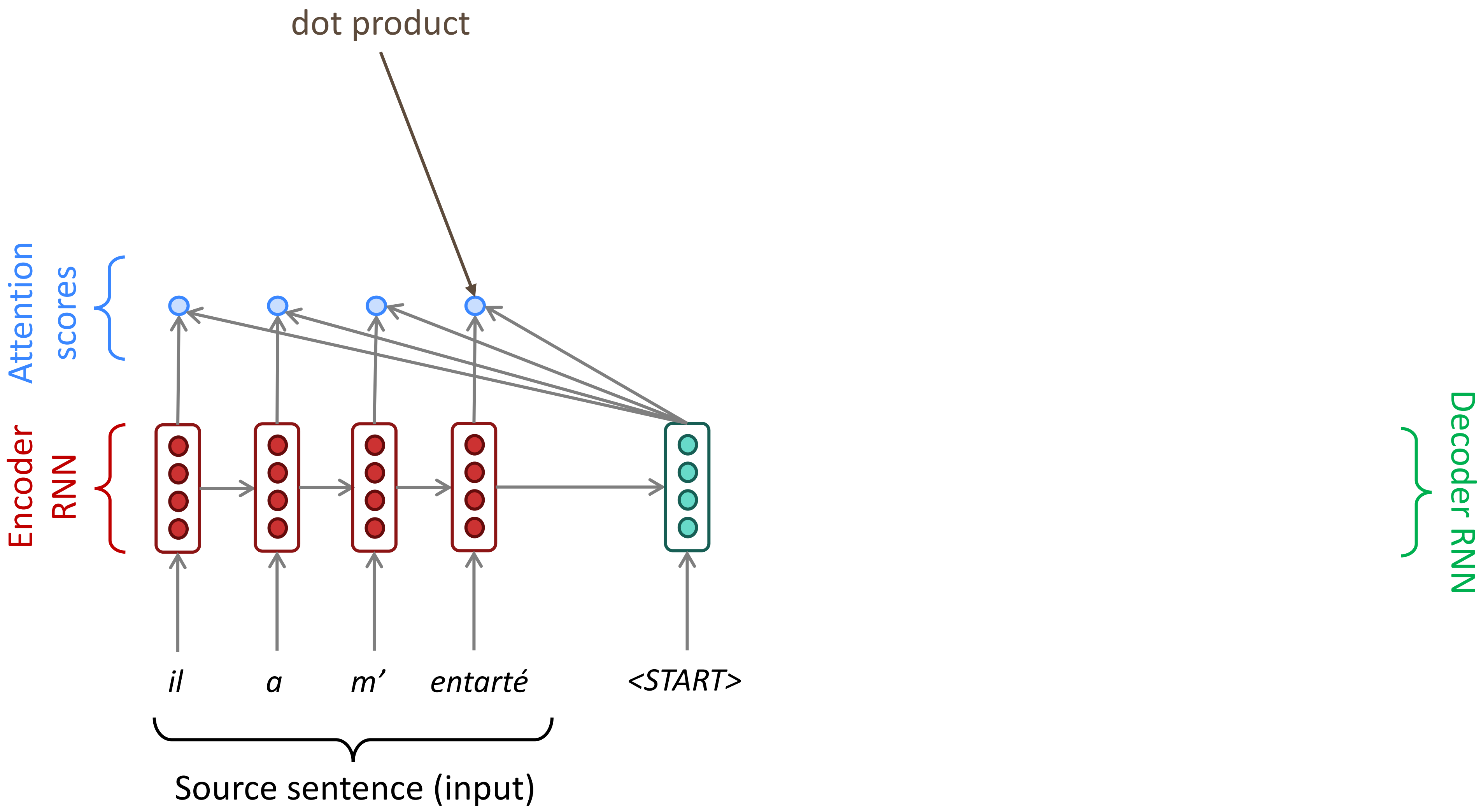
Sequence-to-sequence with attention



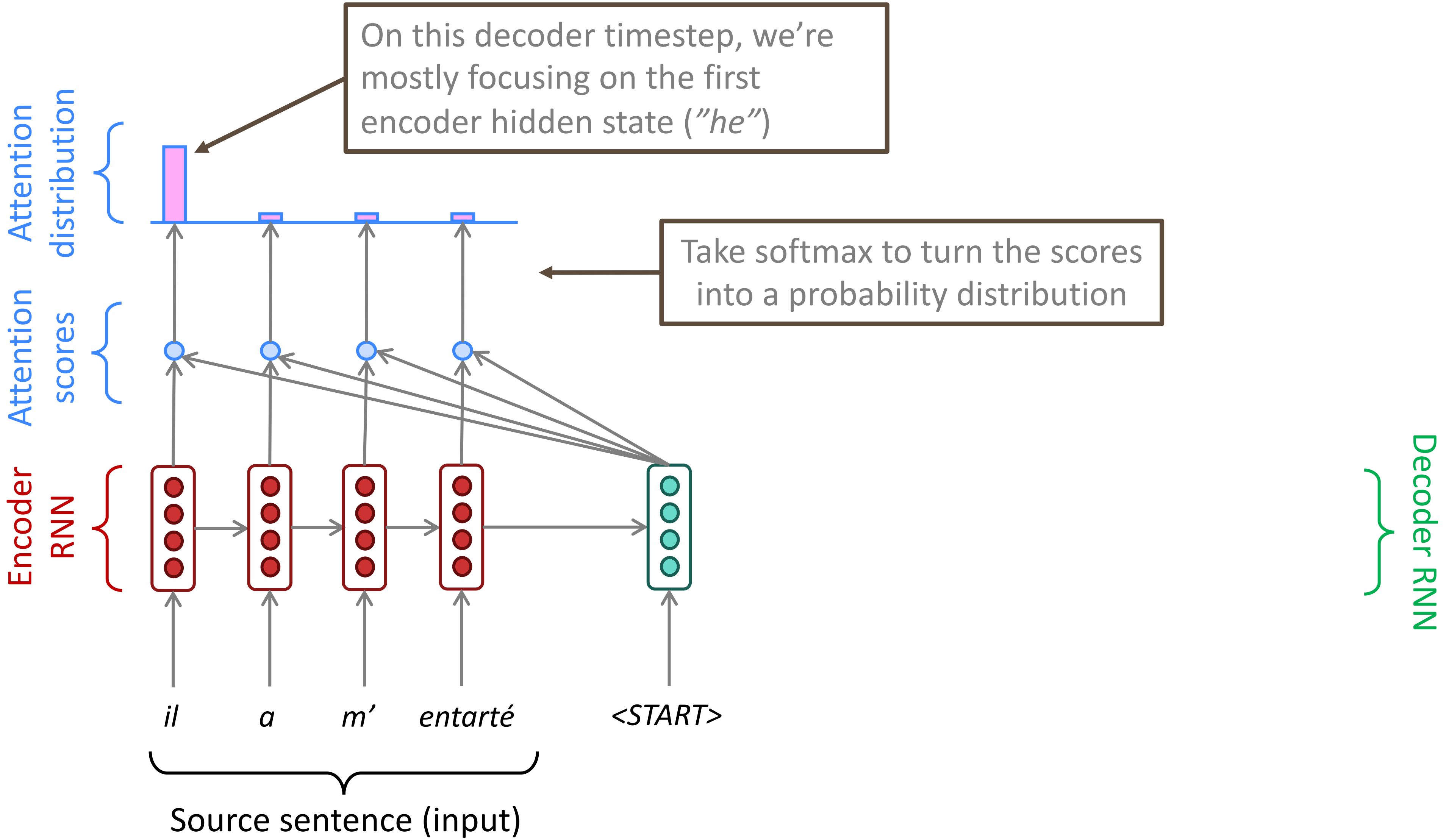
Sequence-to-sequence with attention



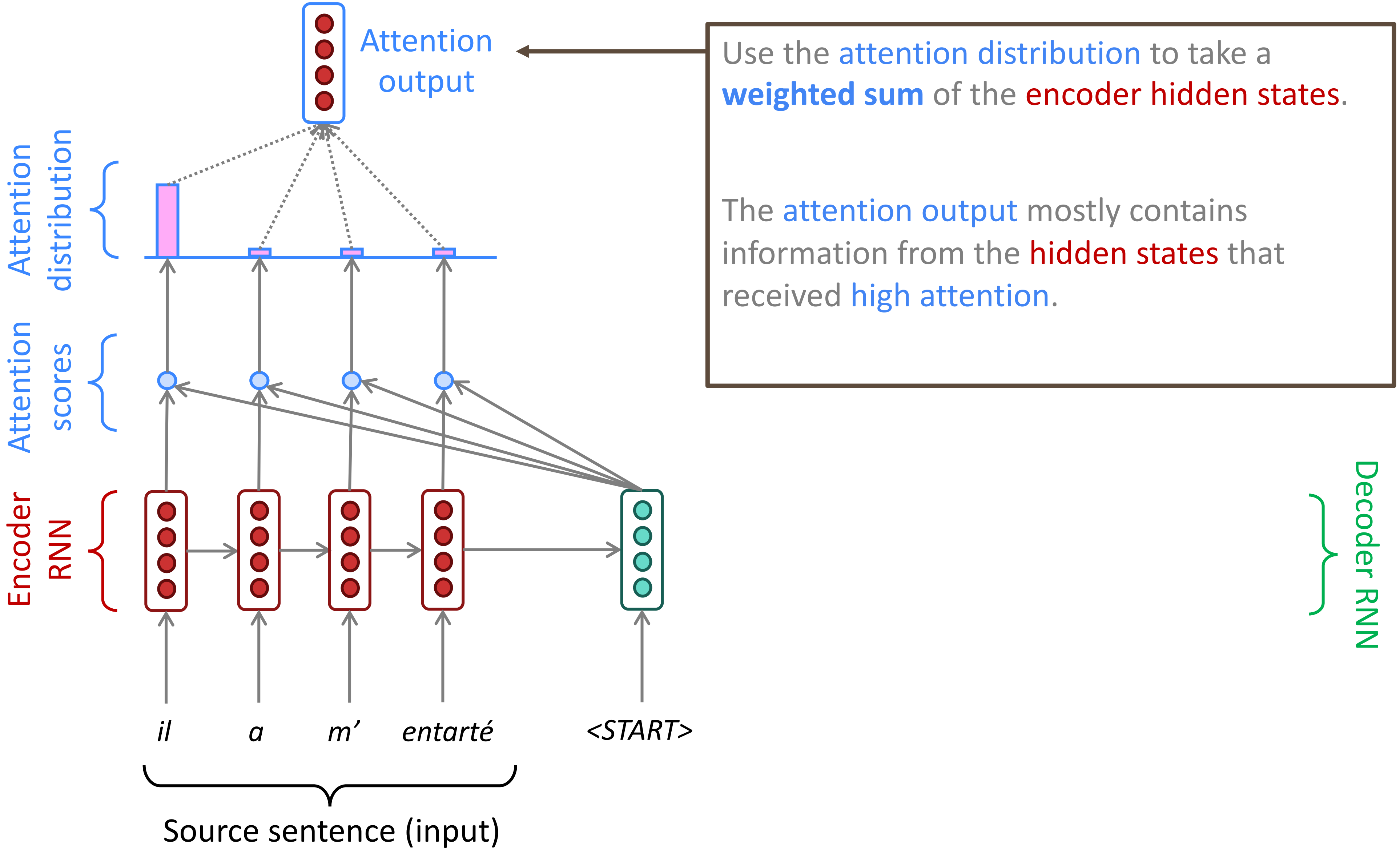
Sequence-to-sequence with attention



Sequence-to-sequence with attention



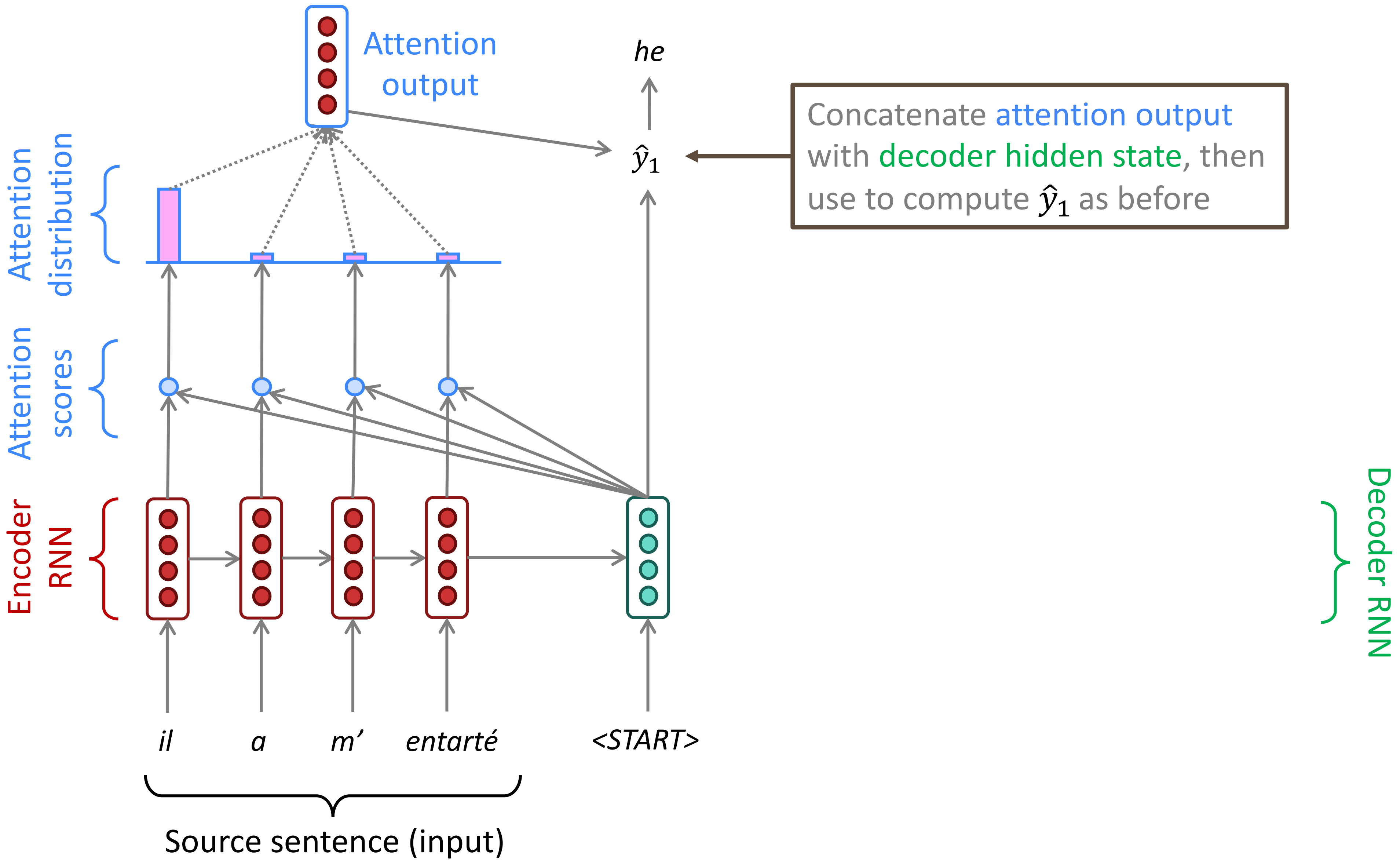
Sequence-to-sequence with attention



Use the attention distribution to take a **weighted sum** of the encoder hidden states.

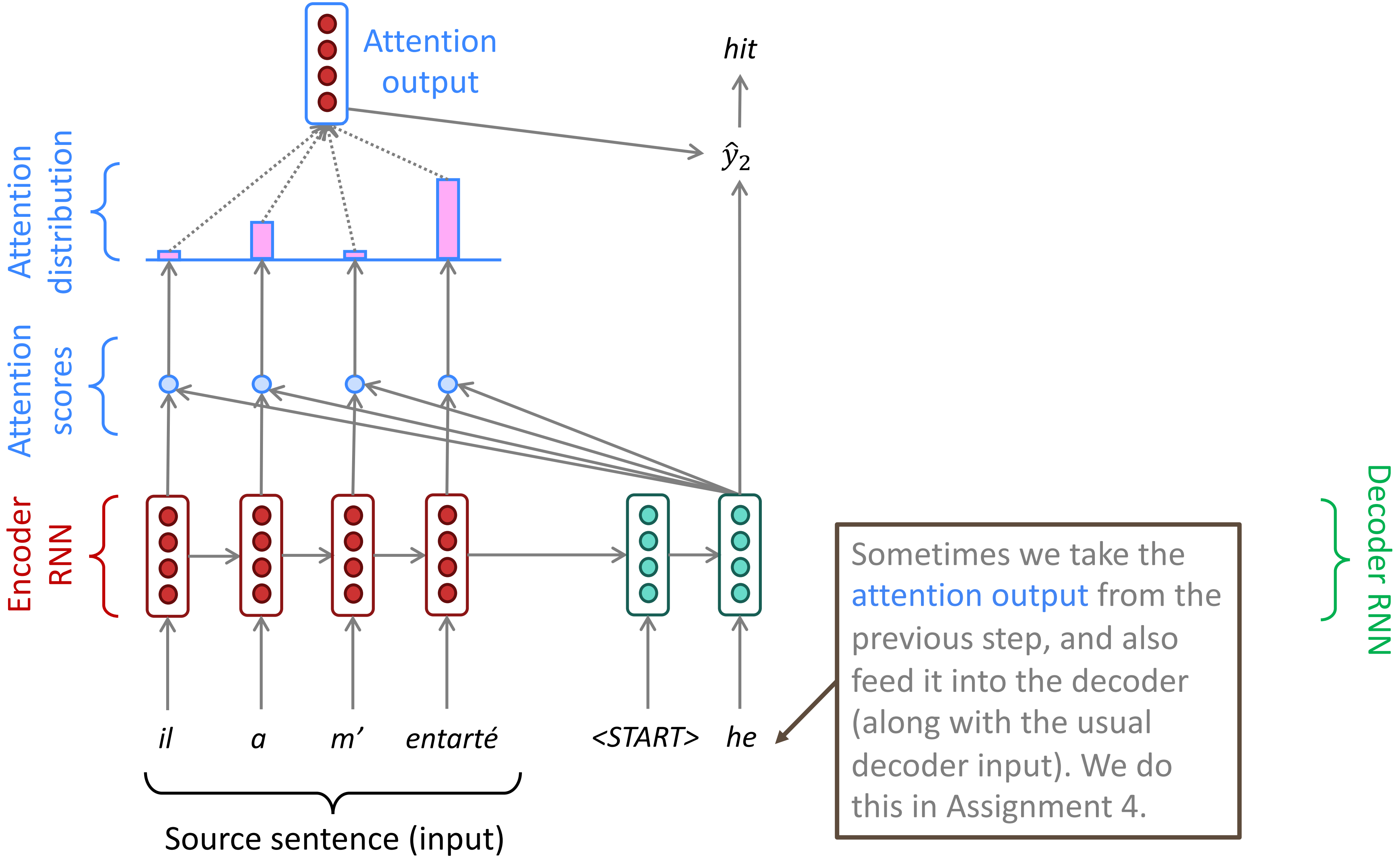
The attention output mostly contains information from the hidden states that received high attention.

Sequence-to-sequence with attention



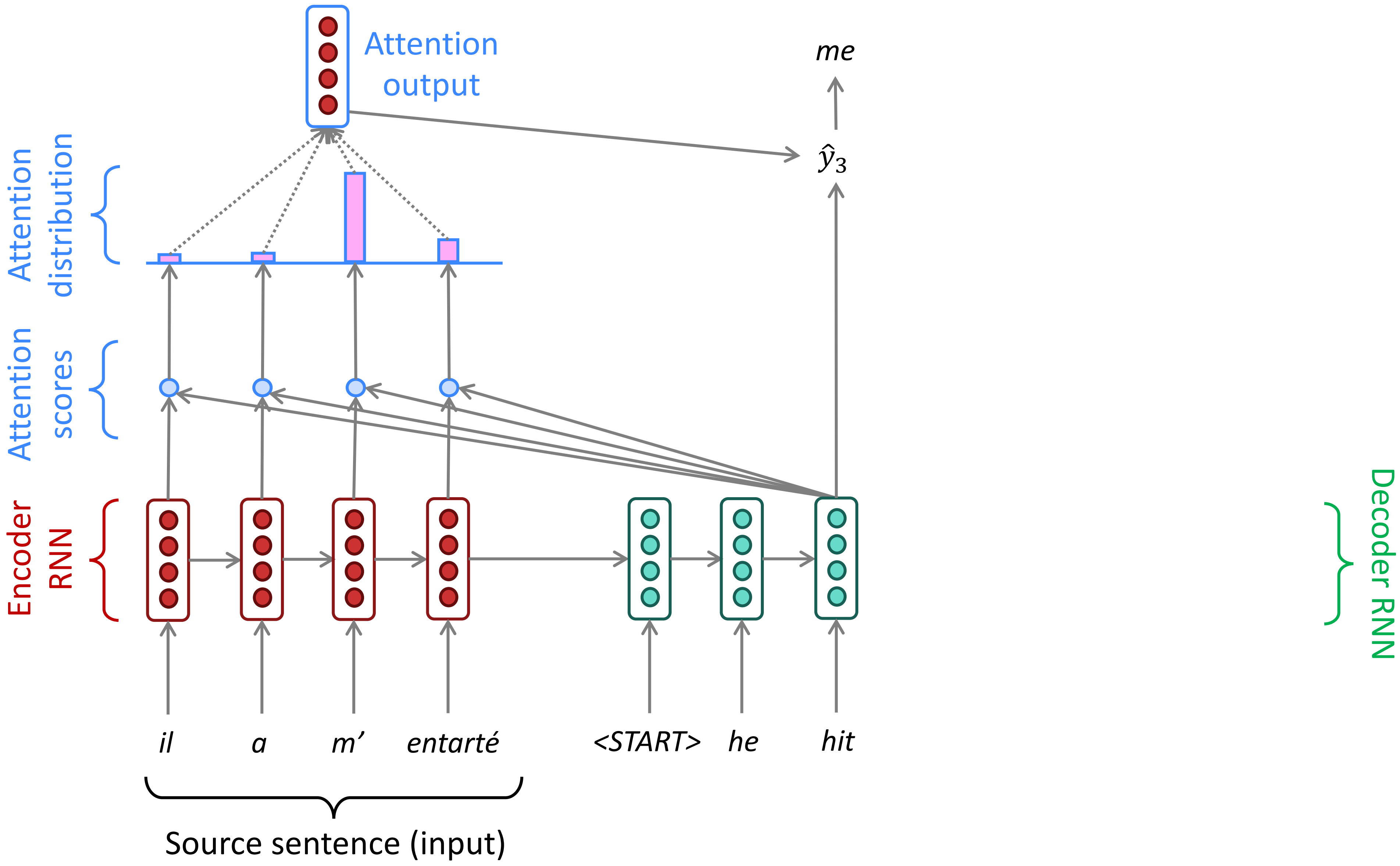
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Sequence-to-sequence with attention

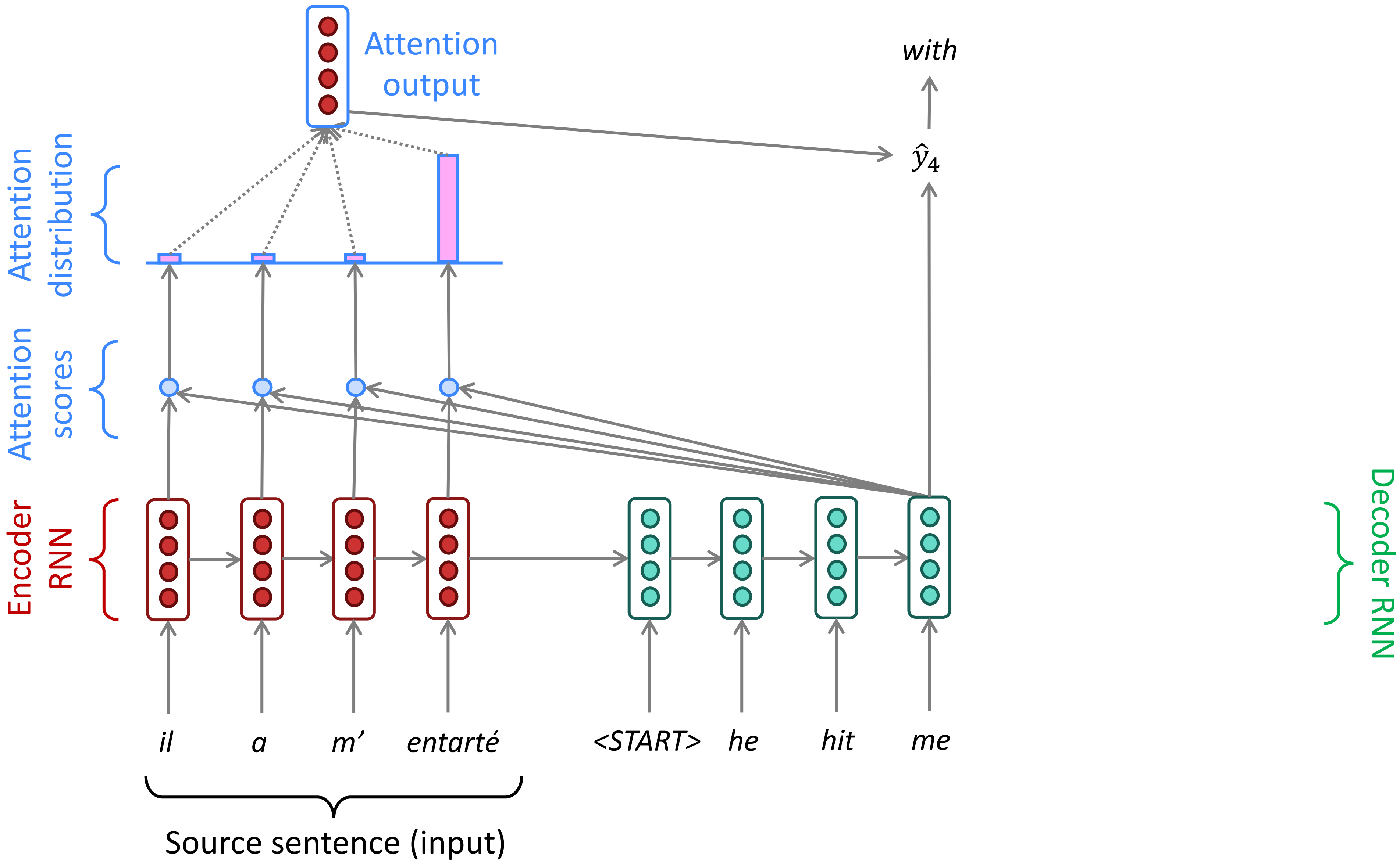


Sometimes we take the attention output from the previous step, and also feed it into the decoder (along with the usual decoder input). We do this in Assignment 4.

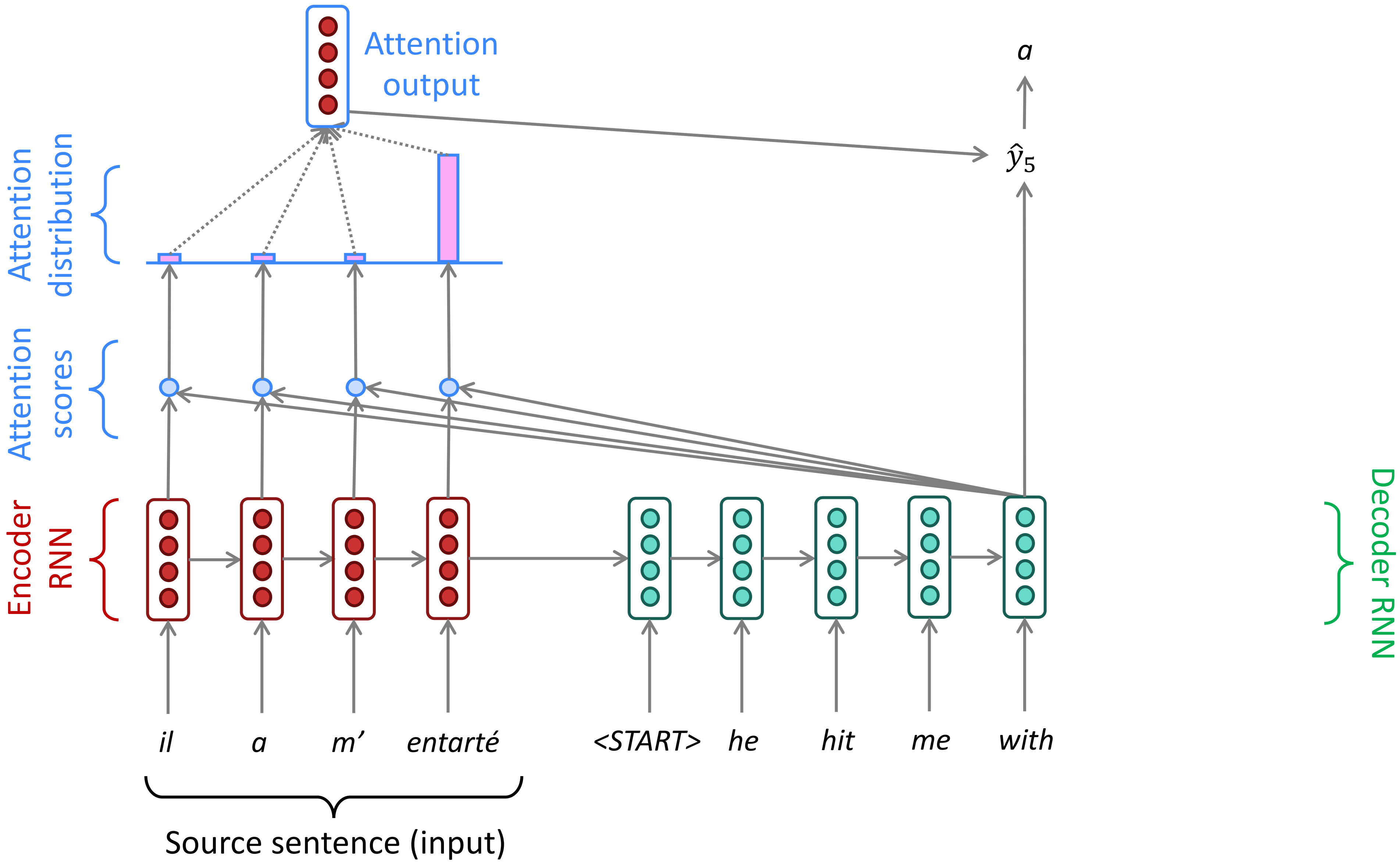
Sequence-to-sequence with attention



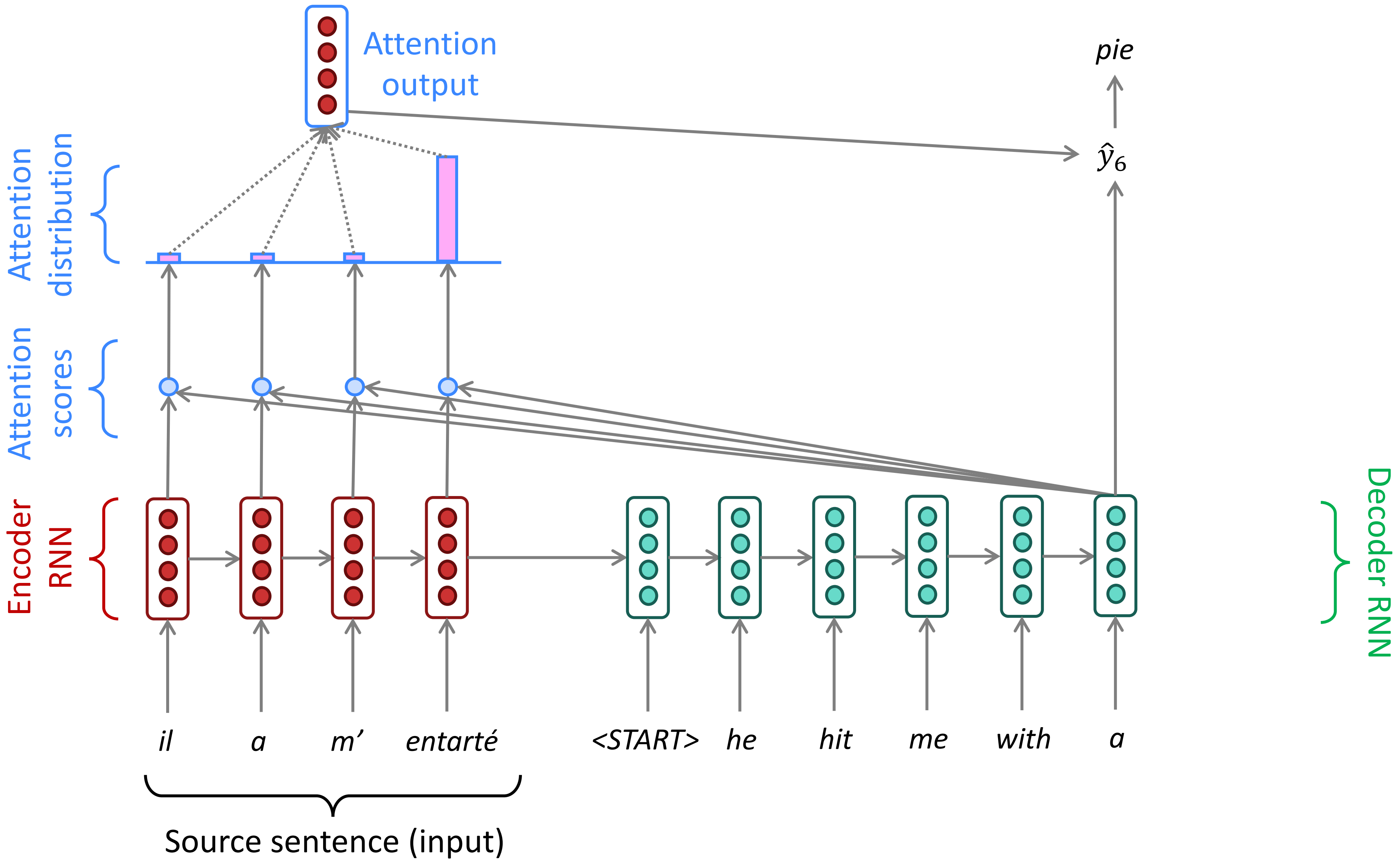
Sequence-to-sequence with attention



Sequence-to-sequence with attention



Sequence-to-sequence with attention



Attention: in equations

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \mathrm{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

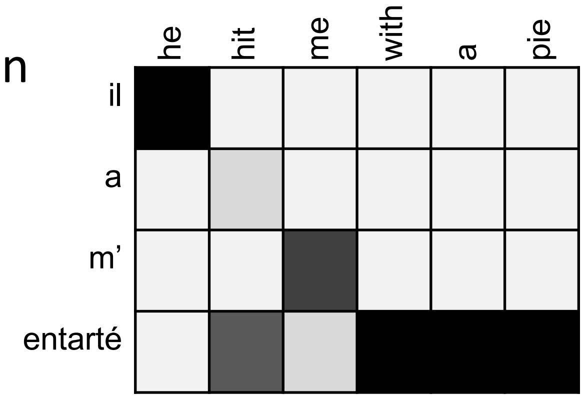
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Attention is great!

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention provides more "human-like" model of the MT process
  - You can look back at the source sentence while translating, rather than needing to remember it all
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with the vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we see what the decoder was focusing on
  - We get (soft) alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself



There are *several* attention variants

- We have some *values* $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

- Attention always involves:
  1. Computing the *attention scores* $\boldsymbol{e} \in \mathbb{R}^N$
  2. Taking softmax to get *attention distribution* α:

  $$\alpha = \mathrm{softmax}(\boldsymbol{e}) \in \mathbb{R}^N$$

  3. Using attention distribution to take weighted sum of values:

  $$\boldsymbol{a} = \sum_{i=1}^{N} \alpha_i \boldsymbol{h}_i \in \mathbb{R}^{d_1}$$

  thus obtaining the *attention output* ***a*** (sometimes called the *context vector*)

There are multiple ways to do this

Attention variants

You'll think about the relative advantages/disadvantages of these in Assignment 4!

There are several ways you can compute $\boldsymbol{e} \in \mathbb{R}^N$ from $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and $\boldsymbol{s} \in \mathbb{R}^{d_2}$:

Basic dot-product attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$

- Note: this assumes $d_1 = d_2$. This is the version we saw earlier.

- Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$ [Luong, Pham, and Manning 2015]
  - Where $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ is a weight matrix. Perhaps better called "bilinear attention"

- Reduced-rank multiplicative attention: $e_i = s^T(\boldsymbol{U}^T\boldsymbol{V})h_i = (\boldsymbol{U}s)^T(\boldsymbol{V}h_i)$
  - For low rank matrices $\boldsymbol{U} \in \mathbb{R}^{k \times d_2}$, $\boldsymbol{V} \in \mathbb{R}^{k \times d_1}$, $k \ll d_1, d_2$

Remember this when we look at Transformers next week!

- Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$ [Bahdanau, Cho, and Bengio 2014]
  - Where $\boldsymbol{W}_1 \in \mathbb{R}^{d_3 \times d_1}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices and $\boldsymbol{v} \in \mathbb{R}^{d_3}$ is a weight vector.
  - d_3 (the attention dimensionality) is a hyperparameter
  - "Additive" is a weird/bad name. It's really using a feed-forward neural net layer.

More information: "Deep Learning for NLP Best Practices", Ruder, 2017. http://ruder.io/deep-learning-nlp-best-practices/index.html#attention
"Massive Exploration of Neural Machine Translation Architectures", Britz et al, 2017, https://arxiv.org/pdf/1703.03906.pdf

Attention is a *general* Deep Learning technique

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)

- **More general definition of attention**:
  - Given a set of vector *values*, and a vector *query*, **attention** is a technique to compute a weighted sum of the values, dependent on the query.

- We sometimes say that the query *attends to* the values.
- For example, in the seq2seq + attention model, each decoder hidden state (query) *attends to* all the encoder hidden states (values).

Attention is a *general* Deep Learning technique

- More general definition of attention:
  - Given a set of vector *values*, and a vector *query*, attention is a technique to compute a weighted sum of the values, dependent on the query.

Intuition:

- The weighted sum is a *selective summary* of the information contained in the values, where the query determines which values to focus on.
- Attention is a way to obtain a *fixed-size representation of an arbitrary set of representations* (the values), dependent on some other representation (the query).

Upshot:

- Attention has become the powerful, flexible, general way pointer and memory manipulation in all deep learning models. A new idea from after 2010! From NMT!

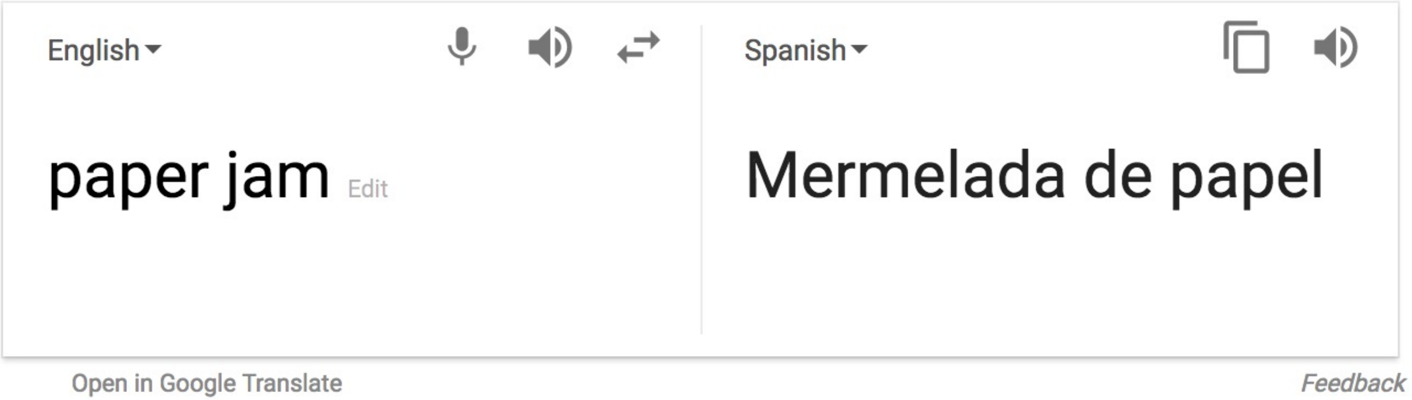
2. So, is Machine Translation solved?

- **Nope!**
- Many difficulties remain:
  - Out-of-vocabulary words
  - Domain mismatch between train and test data
  - Maintaining context over longer text
  - Low-resource language pairs
  - Failures to accurately capture sentence meaning
  - Pronoun (or zero pronoun) resolution errors
  - Morphological agreement errors

Further reading: *"Has AI surpassed humans at translation? Not even close!"*
https://www.skynettoday.com/editorials/state_of_nmt

So is Machine Translation solved?

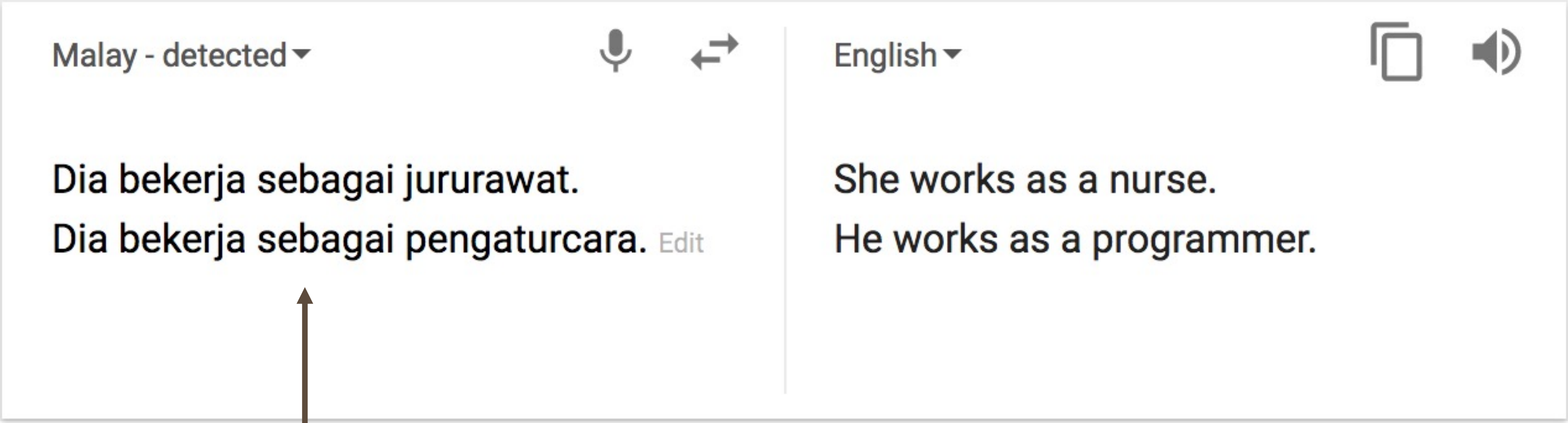
- **Nope!**
- Using common sense is still hard



?

So is Machine Translation solved?

- **Nope!**
- NMT picks up biases in training data



Didn’t specify gender

Source: https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c

So is Machine Translation solved?

TRANSLATE

Reducing gender bias in Google Translate

Dec 06, 2018 · 1 min read

Share

J

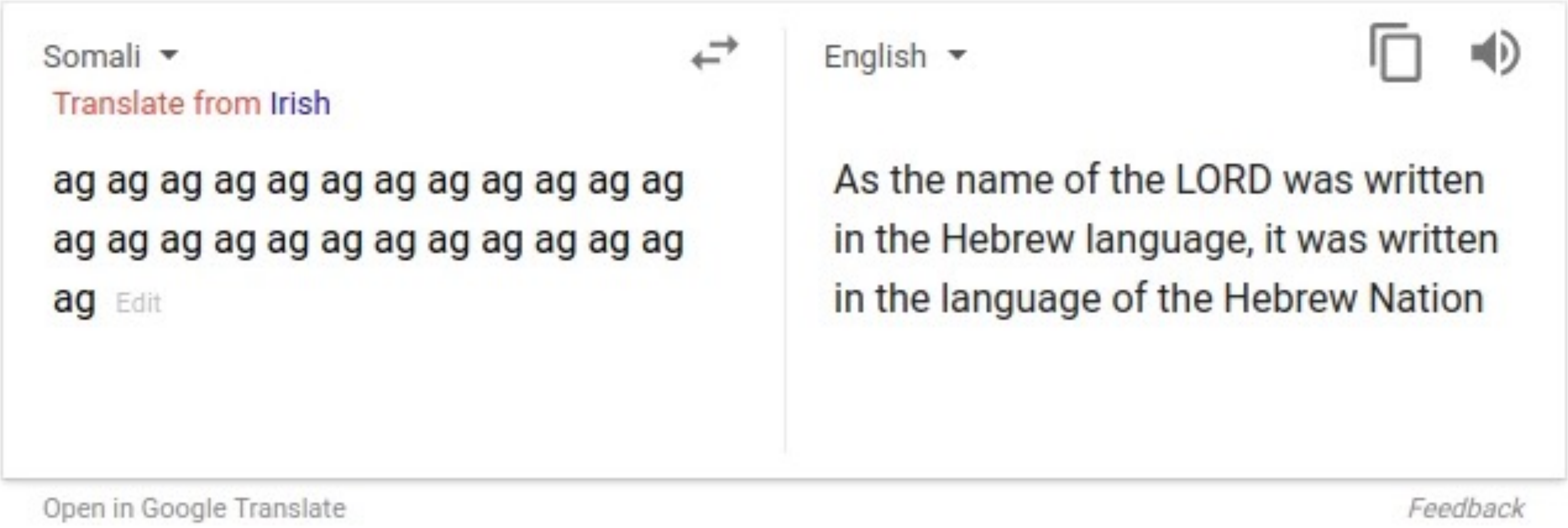
James Kuczmarski
Product Manager, Google Translate

Over the course of this year, there's been an effort across Google to promote fairness and reduce bias in machine learning. Our latest development in this effort addresses gender bias by providing feminine and masculine translations for some gender-neutral words on the Google Translate website.

Source: https://blog.google/products/translate/reducing-gender-bias-google-translate/

So is Machine Translation solved?

- Nope!
- Uninterpretable systems can do strange things
- (But, AFAICS, this problem has been fixed in Google Translate by 2021.)



Picture source: https://www.vice.com/en_uk/article/j5npeg/why-is-google-translate-spitting-out-sinister-religious-prophecies

Explanation: https://www.skynettoday.com/briefs/google-nmt-prophecies

Assignment 4: Cherokee-English machine translation!

- Cherokee is an endangered Native American language – about 2000 fluent speakers
- Extremely low resource: About 20k parallel sentences available, most from the bible
- ᎪᎯᎩᏴ ᏥᎨᏒᎢ ᎦᎵᏉᎩ ᎢᏯᏂᎢ ᎠᏂᏧᏣ. ᏂᎪᎯᎸᎢ ᏗᎦᎳᏫᎢᏍᏗᎢ ᏣᏂᏯᏒᎢ ᏓᎾᏁᎶᎲᏍᎬᎢ ᏅᏯ ᎪᏢᏔᏅᎢ ᎦᏝᏗ ᎠᏂᏐᏝᏧᎵᏙᎲᎢ ᎠᏜ ᎤᏓᏍᏛᏗ ᎦᎾᏍᏗ ᎠᏅᏗᏍᎨᎢ ᎠᏅᏂᎲᎢ.

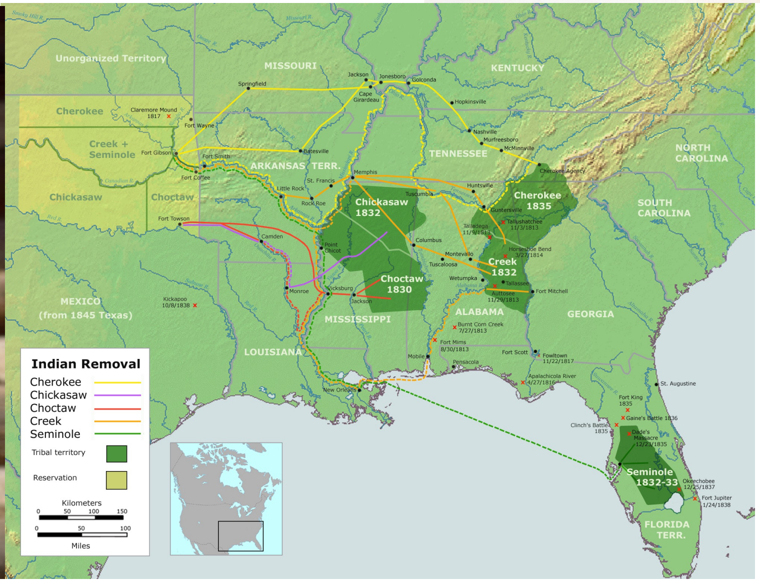
  Long ago were seven boys who used to spend all their time down by the townhouse playing games, rolling a stone wheel along the ground, sliding and striking it with a stick
- Writing system is a syllabary of symbols for each CV unit (85 letters)
- Many thanks to Shiyue Zhang, Benjamin Frey, and Mohit Bansal from UNC Chapel Hill for the resources for this assignment!

- Cherokee is not available on Google Translate! 😭

Cherokee

- Cherokee originally lived in western North Carolina and eastern Tennessee
- Most speakers now in Oklahoma, following the Trail of Tears; some in NC
- Writing system invented by Segwoya (often written Sequoyah) around 1820 – someone who grew up illiterate
  - Very effective: In the following decades Cherokee literacy was higher than for white people in the southeastern United States

- https://www.cherokee.org



NMT research continues

NMT is an important use case for NLP Deep Learning

- NMT research pioneered many of the recent innovations of NLP Deep Learning

- NMT research continues to thrive
  - Researchers have found *many, many* improvements to the “vanilla” seq2seq NMT system we’ve just presented

  - Much work on getting better results on low resource languages

- But, overall, in the last few years more of the excitement has moved to question answering, semantics, inference, natural language generation, ….

3. Course work and grading policy

- 5 x 1-week Assignments: 6% + 4 x 12%: 54%
- Final Default or Custom Course Project (1–3 people): 43%
  - Project proposal: 5%; milestone: 5%; summary paragraph + image: 3%; report: 30%
- Participation: 3%
  - Guest speaker lectures, Ed, our course evals, karma – see website!
- Late day policy
  - 6 free late days; then 1% of total off per day; max 3 late days per assignment
- Collaboration policy: Read the website and the Honor Code!
  - For projects: It's okay to use existing code/resources, but you **must document** it, and you will be graded on your value-add
  - If multi-person: Include a brief statement on the work of each team-mate
    - In almost all cases, each team member gets the same score, but we reserve the right to differentiate in egregious cases

The Final Project

- For FP, you either
  - Do the default project, which is SQuAD question answering (2 sub-variants)
    - Open-ended but an easier start; a good choice for most
  - Propose a custom final project, which we must approve
    - You will receive feedback from a **mentor** (TA/prof/postdoc/PhD)

- You can work in teams of 1–3. Being in a team is encouraged.
  - A larger team project or a project used for multiple classes should be larger and often involves exploring more models or tasks

- You can use any language/framework for your project
  - Though we expect most of you to keep using PyTorch
  - And our starter code for the default FP is in PyTorch

Custom Final Project

- I'm very happy to talk to people about final projects, but the slight problem is that there's only one of me....
- Look at TA expertise for custom final projects:
  - http://web.stanford.edu/class/cs224n/office_hours.html#staff

Mon	**Chris Manning** *Most areas of NLP. Less good on GANs and RL.*				
Mon	**Gaurab Banerjee** *Vision transformers, speech/audio, pretraining*	**Angelica Sun** *NLP, deep learning*	**Lucia Zheng** *NLP, knowledge, LMs for law*	**Vincent Li** *NLP, knowledge, multi-modal*	
Tue	**Kendrick Shen** *Representation learning*	**Sarthak Kanodia** *NLP, CV, data mining, AI for climate change*	**Kamil Ali** *CV, AI for healthcare*	**Yian Zhang** *Pretraining, probing, evaluation, syntax*	
Wed	**Eric Mitchell** *Meta-learning, NLP, continual learning, knowledge editing in LM*	**Ethan A. Chi** *Speech recognition, dialogue systems, interpretability, reasoning*	**Manan Rai** *NLP, Speech, CV*	**Kathy Yu** *ML for health*	
Thu	**Michihiro Yasunaga** *NLP, knowledge*	**Ben Newman** *NLP, evaluation, compositionality, knowledge*	**Kaili Huang** *NLP, dialogue systems*	**Fenglu Hong** *Generative models*	**Anna Goldie** *LM, representation learning, NLP theory, neural net analysis*
Fri	**Grace Lam** *LM, ML for healthcare*	**Allan Zhou** *RL, meta-learning*	**Christopher Wolff**	**Elaine Sui** *ML for healthcare*	

The Default Final Project

- There are two handouts on the web about it now!
- Two variant question answering (QA) tasks
    1. Building a textual question answering architecture for SQuAD from scratch
        - Stanford Question Answering Dataset: https://rajpurkar.github.io/SQuAD-explorer/
        - Provided starter code in PyTorch. ☺ Attempting SQuAD 2.0 (has unanswerable Qs).
    2. Building a Robust QA system which works on different QA datasets/domains
        - You train on SQuAD, NewsQA and Natural Questions; test sets are DuoRC, Race and ZSRE by RC
        - Starting point is large pre-trained LM (DistilBERT); you work mainly on robustness methods
- We will discuss question answering later in the course (week 6). Example:

T: [Bill] Aiken, adopted by Mexican movie actress Lupe Mayorga, grew up in the neighboring town of Madera and his song chronicled the hardships faced by the migrant farm workers he saw as a child.

Q: **In what town did Bill Aiken grow up?**

A: **Madera** [But Google's BERT says <No Answer>!]

Why Choose The Default Final Project?

- If you:
  - Have limited experience with research, don't have any clear idea of what you want to do, or want guidance and a goal, … and a leaderboard, even
- Then:
  - Do the default final project!
  - Many people should do it! (Past statistics: about half of people do DFP.)

- Considerations:
  - The two default final project variants give you lots of guidance, scaffolding, and clear goalposts to aim at
  - The path to success is not to do something that looks kinda weak compared to what you could have done with the DFP.

Why Choose The Custom Final Project?

- If you:
  - Have some research project that you're excited about (and are possibly already working on), **which substantively involves human language and neural networks**
  - You want to try to do something different on your own
  - You're just interested in something other than question answering (that involves human language material and deep learning)
  - You want to see more of the process of defining a research goal, finding data and tools, and working out something you could do that is interesting, and how to evaluate it
- Then:
  - Do the custom final project!

Gamesmanship

- The default final projects are a more guided option, but it’s not that they’re a less work option
- The default final projects are also open-ended projects where you can explore different approaches, but to a given problem. Strong default final projects do this.
- There are great default final projects and great custom final projects … and there are weak default final projects and weak custom final projects. It’s not that either option is the easy way to get a good grade
- We give Best Project Awards for both default and custom final projects

Project Proposal – from every team 5%

1. Find a relevant (key) research paper for your topic
   - For DFP, we provide some suggestions, but you might look elsewhere for interesting QA/reading comprehension work
2. Write a summary of that research paper and what you took away from it as key ideas that you hope to use
3. Write what you plan to work on and how you can innovate in your final project work
   - Suggest a good milestone to have achieved as a halfway point
4. Describe as needed, especially for Custom projects:
   - A project plan, relevant existing literature, the kind(s) of models you will use/explore; the **data** you will use (and how it is obtained), and how you will **evaluate** success

3–4 pages, due Tue Feb 8, 3:15pm on Gradescope

Project Proposal – from everyone 5%

2. Skill: How to think critically about a research paper
   - What were the main novel contributions or points?
   - Is what makes it work something general and reusable or a special case?
   - Are there flaws or neat details in what they did?
   - How does it fit with other papers on similar topics?
   - Does it provoke good questions on further or different things to try?
     - Grading of research paper review is primarily summative
3. How to do a good job on your project plan
   - You need to have an overall sensible idea (!)
   - But most project plans that are lacking are lacking in nuts-and-bolts ways:
     - Do you have appropriate data or a realistic plant to be able to collect it in a short period of time
     - Do you have a realistic way to evaluate your work
     - Do you have appropriate baselines or proposed ablation studies for comparisons
   - Grading of project proposal is primarily formative

Project Milestone – from everyone 5%

- This is a progress report
- You should be more than halfway done!
- Describe the experiments you have run
- Describe the preliminary results you have obtained
- Describe how you plan to spend the rest of your time

You are expected to **have implemented some system** and to **have some initial experimental results** to show by this date (except for certain unusual kinds of projects)

Due Thu Feb 24, 3:15pm on Gradescope

Project writeup

- Writeup quality is very important to your grade!!!
  - Look at recent years' prize winners for examples

Abstract Introduction	Prior related work	Model	Model
Data	Experiments	Results	Analysis & Conclusion

4. Finding Research Topics

Two basic starting points, for all of science:

- [Nails] Start with a (domain) problem of interest and try to find good/better ways to address it than are currently known/used
- [Hammers] Start with a technical method/approach of interest, and work out good ways to extend it, improve it, understand it, or find new ways to apply it

Project types

This is not an exhaustive list, but most projects are one of

1. Find an application/task of interest and explore how to approach/solve it effectively, often with an existing model
   - Could be a task in the wild or some existing Kaggle/bake-off/shared task
2. Implement a complex neural architecture and demonstrate its performance on some data
3. Come up with a new or variant neural network model or approach and explore its empirical success
4. Analysis project. Analyze the behavior of a model: how it represents linguistic knowledge or what kinds of phenomena it can handle or errors that it makes
5. Rare theoretical project: Show some interesting, non-trivial properties of a model type, data, or a data representation

Deep Poetry: Word-Level and Character-Level Language Models for Shakespearean Sonnet Generation

Stanley Xie, Ruchir Rastogi and Max Chang

<u>Gated LSTM</u>

Thy youth 's time and face his form shall cover?
Now all fresh beauty, my love there
Will ever Time to greet, forget each, like ever decease,
But in a best at worship his glory die.

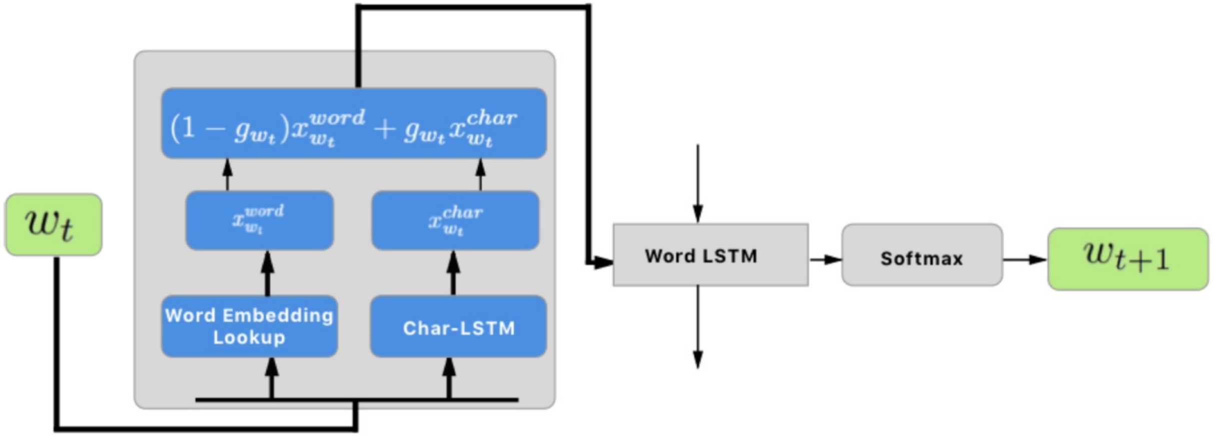
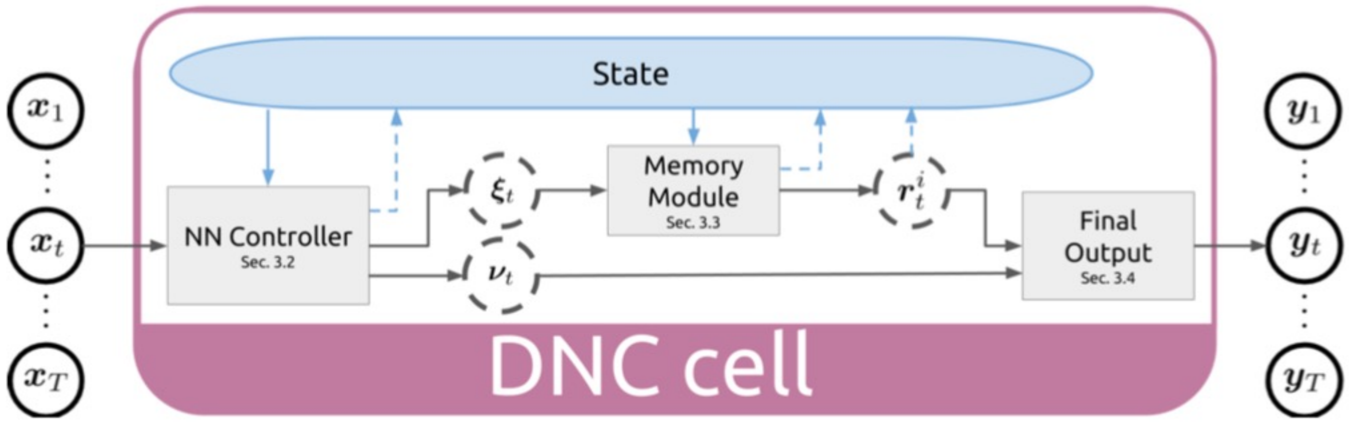


Figure 1: Architecture of the Gated LSTM

Implementation and Optimization of Differentiable Neural Computers

Carol Hsin
Graduate Student in Computational & Mathematical Engineering

We implemented and optimized Differentiable Neural Computers (DNCs) as described in the Oct. 2016 DNC paper [1] on the bAbI dataset [25] and on copy tasks that were described in the Neural Turning Machine paper [12]. This paper will give the reader a better understanding of this new and promising architecture through the documentation of the approach in our DNC implementation and our experience of the challenges of optimizing DNCs.



Improved Learning through Augmenting the Loss

Hakan Inan
inanh@stanford.edu

Khashayar Khosravi
khosravi@stanford.edu

We present two improvements to the well-known Recurrent Neural Network Language Models(RNNLM). First, we use the word embedding matrix to project the RNN output onto the output space and already achieve a large reduction in the number of free parameters while still improving performance. Second, instead of merely minimizing the standard cross entropy loss between the prediction distribution and the "one-hot" target distribution, we minimize an additional loss term which takes into account the inherent metric similarity between the target word and other words. We show with experiments on the Penn Treebank Dataset that our proposed model (1) achieves significantly lower average word perplexity than previous models with the same network size and (2) achieves the new state of the art by using much fewer parameters than used in the previous best work.

Published as a conference paper at ICLR 2017

Word2Bits - Quantized Word Vectors

Maximilian Lam
maxlam@stanford.edu

Abstract

Word vectors require significant amounts of memory and storage, posing issues to resource limited devices like mobile phones and GPUs. We show that high quality quantized word vectors using 1-2 bits per parameter can be learned by introducing a quantization function into Word2Vec. We furthermore show that training with the quantization function acts as a regularizer. We train word vectors on English Wikipedia (2017) and evaluate them on standard word similarity and analogy tasks and on question answering (SQuAD). Our quantized word vectors not only take 8-16x less space than full precision (32 bit) word vectors but also outperform them on word similarity tasks and question answering.

How to find an interesting place to start?

- Look at ACL anthology for NLP papers:
  - https://aclanthology.org/
- Also look at the online proceedings of major ML conferences:
  - NeurIPS https://papers.nips.cc, ICML, ICLR https://openreview.net/group?id=ICLR.cc
- Look at past cs224n projects
  - See the class website
- Look at online preprint servers, especially:
  - https://arxiv.org

- Even better: look for an interesting problem in the world!
  - Hal Varian: How to Build an Economic Model in Your Spare Time
    https://people.ischool.berkeley.edu/~hal/Papers/how.pdf

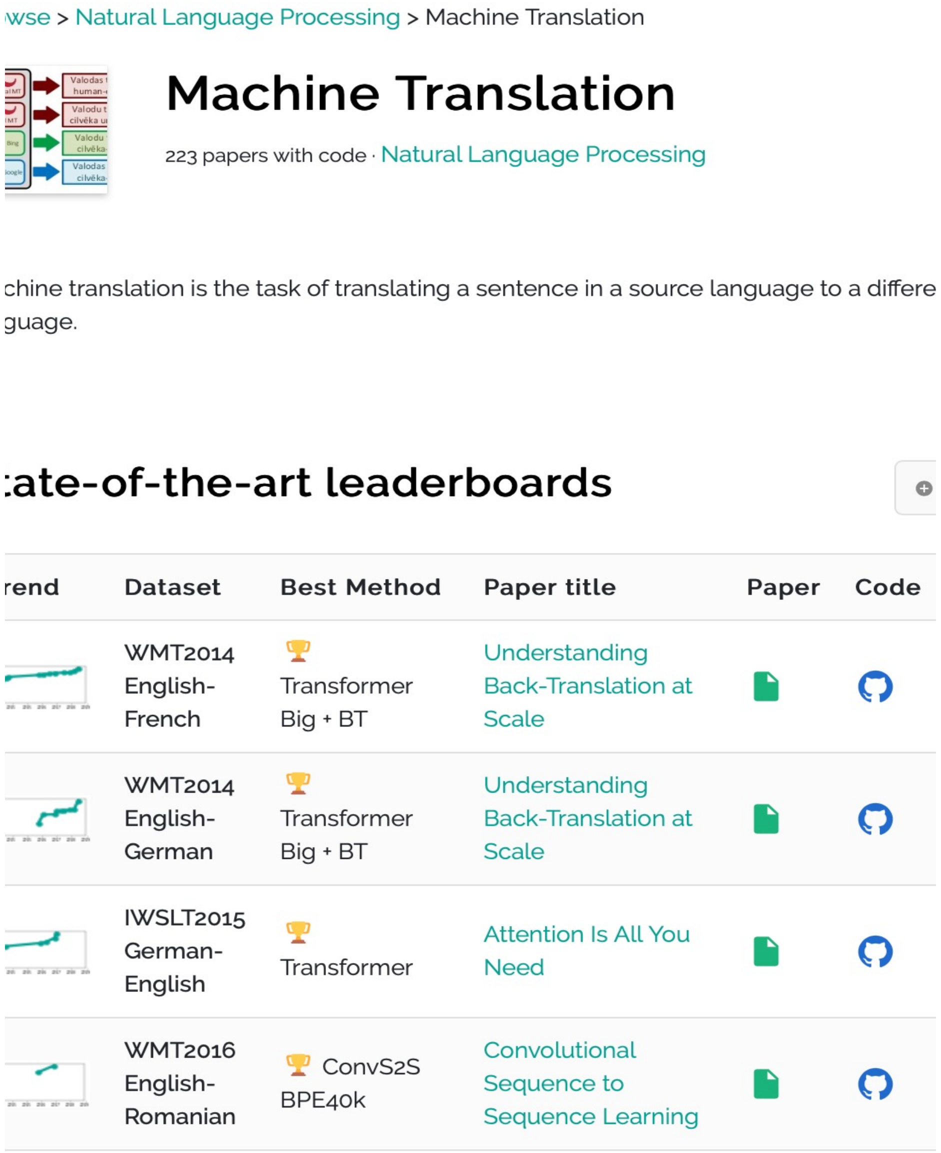
Want to beat the state of the art on something?

Great new sites that try to collate info on the state of the art

- Not always correct, though

https://paperswithcode.com/sota
https://nlpprogress.com/

Specific tasks/topics. Many, e.g.:
https://gluebenchmark.com/leaderboard/
https://www.conll.org/previous-tasks/


wse > Natural Language Processing > Machine Translation

Machine Translation

223 papers with code · Natural Language Processing

chine translation is the task of translating a sentence in a source language to a differe guage.

:ate-of-the-art leaderboards

rend	Dataset	Best Method	Paper title	Paper	Code
	WMT2014 English-French	Transformer Big + BT	Understanding Back-Translation at Scale		
	WMT2014 English-German	Transformer Big + BT	Understanding Back-Translation at Scale		
	IWSLT2015 German-English	Transformer	Attention Is All You Need		
	WMT2016 English-Romanian	ConvS2S BPE40k	Convolutional Sequence to Sequence Learning		

Finding a topic

- Turing award winner and Stanford CS emeritus professor Ed Feigenbaum says to follow the advice of his advisor, AI pioneer, and Turing and Nobel prize winner Herb Simon:
  - “If you see a research area where many people are working, go somewhere else.”
- But where to go? Wayne Gretzky:
  - “I skate to where the puck is going, not where it has been.”

Old Deep Learning (NLP), new Deep Learning NLP

- In the early days of the Deep Learning revival (2010-2018), most of the work was in defining and exploring better deep learning architectures
- Typical paper:
  - I can improve a summarization system by not only using attention standardly, but allowing copying attention – where you use additional attention calculations and an additional probabilistic gate to simply copy a word from the input to the output
- That's what a lot of good CS 224N projects did too

- In 2019–2022, that approach is dead
  - Well, that's too strong, but it's difficult and much rarer

- Most work downloads a big pre-trained model (which fixes the architecture)
  - Action is in fine-tuning, or domain adaptation followed by fine-tuning, etc., etc.

2022 NLP … recommended for all your practical projects ☺

pip install transformers # By Huggingface 🤗

not quite runnable code but gives the general idea….

from transformers import BertForSequenceClassification, AutoTokenizer

model = BertForSequenceClassification.from_pretrained('bert-base-uncased’)

model.train()

tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased’)

fine_tuner = Trainer(model=model, args=training_args, train_dataset=train_dataset, eval_dataset=test_dataset)

fine_tuner.train()

eval_dataset = load_and_cache_examples(args, eval_task, tokenizer, evaluate=True)

results = evaluate(model, tokenizer, eval_dataset, args)

Exciting areas 2022

A lot of what is exciting now is problems that work within or around this world

- Evaluating and improving models for something other than accuracy
  - Robustness to domain shift
  - Evaluating the robustness of models in general (someone could hack on this new project as their final project!): https://robustnessgym.com
- Doing empirical work looking at what large pre-trained models have learned
- Working out how to get knowledge and good task performance from large models for particular tasks without much data (transfer learning, etc.)
- Looking at the bias, trustworthiness, and explainability of large models
- Working on how to augment the data for models to improve performance
- Looking at low resource languages or problems
- Improving performance on the tail of rare stuff, addressing bias

Exciting areas 2022

- Scaling models up and down
  - Building big models is BIG: GPT-2 and GPT-3 … **but just not possible for a cs224n project** – do also be realistic about the scale of compute you can do!
  - Building small, performant models is also BIG. This could be a great project
    - Model pruning, e.g.: https://papers.nips.cc/paper/2020/file/eae15aabaa768ae4a5993a8a4f4fa6e4-Paper.pdf
    - Model quantization, e.g.: https://arxiv.org/pdf/2004.07320.pdf
    - How well can you do QA in 6GB or 500MB? https://efficientqa.github.io
- Looking to achieve more advanced functionalities
  - E.g., compositionality, systematic generalization, fast learning (e.g., meta-learning) on smaller problems and amounts of data, and more quickly
    - BabyAI: https://arxiv.org/abs/2007.12770
    - gSCAN: https://arxiv.org/abs/2003.05161

5. Finding data

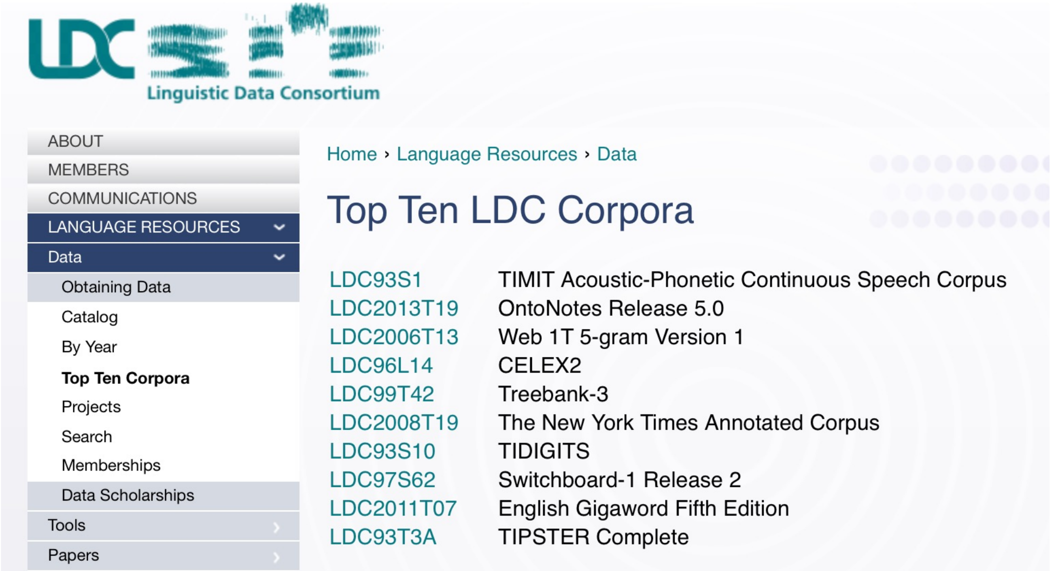
- Some people collect their own data for a project – **we like that!**
  - You may have a project that uses “unsupervised” data
  - You can annotate a small amount of data
  - You can find a website that effectively provides annotations, such as likes, stars, ratings, responses, etc.
    - Let’s you learn about real word challenges of applying ML/NLP!
  - **But be careful on scoping things so that this doesn’t take most of your time!!!**

- Some people have existing data from a research project or company
  - Fine to use providing you can provide data samples for submission, report, etc.

- **Most people make use of an existing, curated dataset built by previous researchers**
  - You get a fast start and there is obvious prior work and baselines

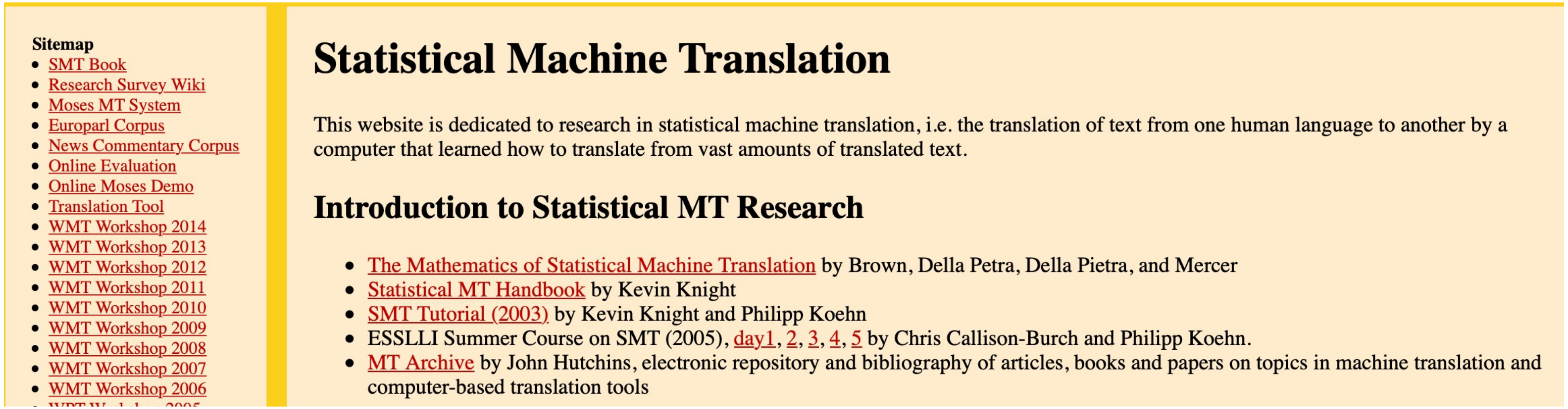
Linguistic Data Consortium

- https://catalog.ldc.upenn.edu/
- Stanford licenses data; you can get access by signing up at: https://linguistics.stanford.edu/resources/resources-corpora
- Treebanks, named entities, coreference data, lots of clean newswire text, lots of speech with transcription, parallel MT data, etc.
  - Look at their catalog
  - Don't use for non-Stanford purposes!



Machine translation

- http://statmt.org
- Look in particular at the various WMT shared tasks



Sitemap
- SMT Book
- Research Survey Wiki
- Moses MT System
- Europarl Corpus
- News Commentary Corpus
- Online Evaluation
- Online Moses Demo
- Translation Tool
- WMT Workshop 2014
- WMT Workshop 2013
- WMT Workshop 2012
- WMT Workshop 2011
- WMT Workshop 2010
- WMT Workshop 2009
- WMT Workshop 2008
- WMT Workshop 2007
- WMT Workshop 2006

Statistical Machine Translation

This website is dedicated to research in statistical machine translation, i.e. the translation of text from one human language to another by a computer that learned how to translate from vast amounts of translated text.

Introduction to Statistical MT Research

- The Mathematics of Statistical Machine Translation by Brown, Della Petra, Della Pietra, and Mercer
- Statistical MT Handbook by Kevin Knight
- SMT Tutorial (2003) by Kevin Knight and Philipp Koehn
- ESSLLI Summer Course on SMT (2005), day1, 2, 3, 4, 5 by Chris Callison-Burch and Philipp Koehn.
- MT Archive by John Hutchins, electronic repository and bibliography of articles, books and papers on topics in machine translation and computer-based translation tools

Dependency parsing: Universal Dependencies

- https://universaldependencies.org

Universal Dependencies

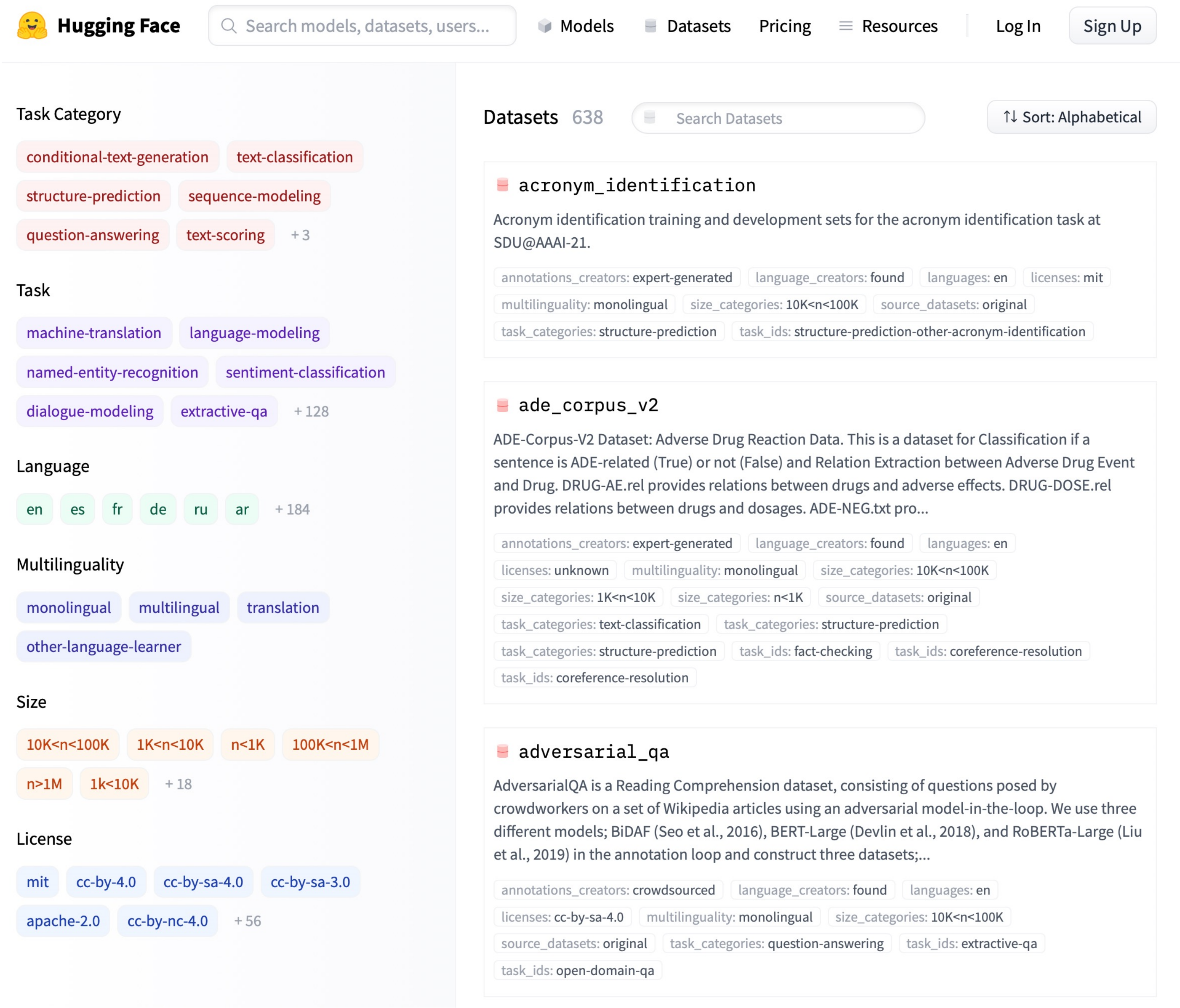
Universal Dependencies (UD) is a framework for cross-linguistically consistent grammatical annotation and an open community effort with over 200 contributors producing more than 100 treebanks in over 70 languages.

- Short introduction to UD
- UD annotation guidelines
- More information on UD:
  - How to contribute to UD
  - Tools for working with UD
  - Discussion on UD
  - UD-related events
- Query UD treebanks online:
  - SETS treebank search maintained by the University of Turku
  - PML Tree Query maintained by the Charles University in Prague
  - Kontext maintained by the Charles University in Prague
  - Grew-match maintained by Inria in Nancy
- Download UD treebanks

If you want to receive news about Universal Dependencies, you can subscribe to the UD mailing list. If you want to discuss individual annotation questions, use the Github issue tracker.

🤗 Huggingface Datasets

- https://huggingface.co/datasets



Paperswithcode Datasets

- https://www.paperswithcode.com/datasets?mod=texts&page=1

835 dataset results for Texts ×

Penn Treebank
The English Penn Treebank corpus, and in particular the section of the corpus corresponding to the articles of Wall Street Journal (WSJ), is one of the most known and used corpus for t...
1,545 PAPERS • 10 BENCHMARKS

SQuAD (Stanford Question Answering Dataset)
The Stanford Question Answering Dataset (SQuAD) is a collection of question-answer pairs derived from Wikipedia articles. In SQuAD, the correct answers of questions can be any se-...
1,254 PAPERS • 7 BENCHMARKS

Visual Genome
Visual Genome contains Visual Question Answering data in a multi-choice setting. It consists of 101,174 images from MSCOCO with 1.7 million QA pairs, 17 questions per image on aver-...
903 PAPERS • 11 BENCHMARKS

GLUE

GLUE (General Language Understanding Evaluation benchmark)
General Language Understanding Evaluation (GLUE) benchmark is a collection of nine natural language understanding tasks, including single-sentence tasks CoLA and SST-2, similarity...
847 PAPERS • 14 BENCHMARKS

SNLI (Stanford Natural Language Inference)
The SNLI dataset (Stanford Natural Language Inference) consists of 570k sentence-pairs manually labeled as entailment, contradiction, and neutral. Premises are image captions fro...
743 PAPERS • 1 BENCHMARK

CLEVR (Compositional Language and Elementary Visual Reasoning)
CLEVR (Compositional Language and Elementary Visual Reasoning) is a synthetic Visual Question Answering dataset. It contains images of 3D-rendered objects; each image comes...
528 PAPERS • 1 BENCHMARK

Visual Question Answering (VQA)
Visual Question Answering (VQA) is a dataset containing open-ended questions about images. These questions require an understanding of vision, language and commonsense...
435 PAPERS • 2 BENCHMARKS

BILLION WORD BENCHMARK

Billion Word Benchmark
The One Billion Word dataset is a dataset for language modeling. The training/held-out data was produced from the WMT 2011 News Crawl data using a combination of Bash shell and...
417 PAPERS • 1 BENCHMARK

Many, many more

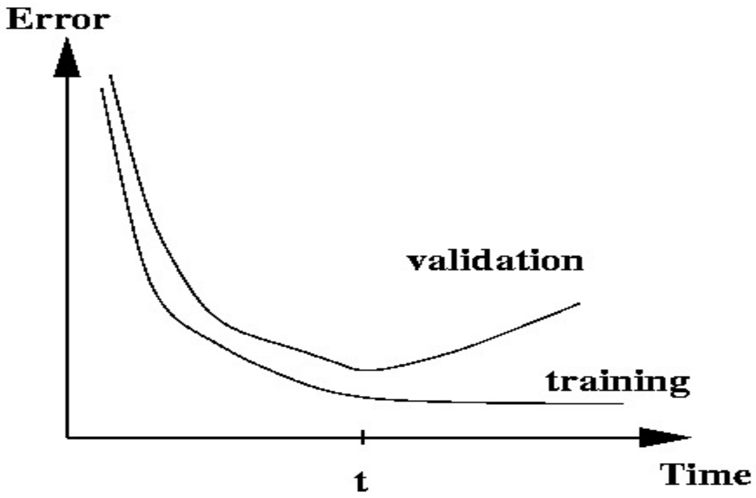
- There are now many other datasets available online for all sorts of purposes
  - Look at Kaggle
  - Look at research papers to see what data they use
  - Look at lists of datasets
    - https://machinelearningmastery.com/datasets-natural-language-processing/
    - https://github.com/niderhoff/nlp-datasets
  - Lots of particular things:
    - https://gluebenchmark.com/tasks
    - https://nlp.stanford.edu/sentiment/
    - https://research.fb.com/downloads/babi/ (Facebook bAbI-related)
  - Ask on Ed or talk to course staff

6. Care with datasets and in model development

- Many publicly available datasets are released with a **train/dev/test** structure.
- **We're all on the honor system to do test-set runs only when development is complete.**
- Splits like this presuppose a fairly large dataset.
- If there is no dev set or you want a separate tune set, then you create one by splitting the training data
  - We have to weigh the usefulness of it being a certain size against the reduction in train-set size.
  - Cross-validation (q.v.) is a technique for maximizing data when you don't have much
- Having a fixed test set ensures that all systems are assessed against the same gold data. This is generally good, but it is problematic when the test set turns out to have unusual properties that distort progress on the task.

Training models and pots of data

- When training, models **overfit** to what you are training on
  - The model correctly describes what happened to occur in particular data you trained on, but the patterns are not general enough patterns to be likely to apply to new data
- The way to monitor and avoid problematic overfitting is using **independent** validation and test sets …



Training models and pots of data

- You build (estimate/train) a model on a **training set**.
- Often, you then set further hyperparameters on another, independent set of data, the **tuning set**
  - The tuning set is the training set for the hyperparameters!
- You measure progress as you go on a **dev set** (development test set or validation set)
  - If you do that a lot you overfit to the dev set so it can be good to have a second dev set, the **dev2** set
- **Only at the end**, you evaluate and present final numbers on a **test set**
  - Use the final test set **extremely** few times … ideally only once

Training models and pots of data

- The **train**, **tune**, **dev**, and **test** sets need to be completely distinct
- It is invalid to give results testing on material you have trained on
  - You will get a falsely good performance.
  - We almost always overfit on train
- You need an independent tuning set
  - The hyperparameters won't be set right if tune is same as train
- If you keep running on the same evaluation set, you begin to overfit to that evaluation set
  - Effectively you are "training" on the evaluation set … you are learning things that do and don't work on that particular eval set and using the info
- To get a valid measure of system performance you need another untrained on, **independent** test set … hence dev2 and final test

Getting your neural network to train

- Start with a positive attitude!
  - **Neural networks want to learn!**
    - If the network isn't learning, you're doing something to prevent it from learning successfully

- Realize the grim reality:
  - **There are lots of things that can cause neural nets to not learn at all or to not learn very well**
    - Finding and fixing them ("debugging and tuning") can often take more time than implementing your model

- It's hard to work out what these things are
  - But experience, experimental care, and rules of thumb help!

Experimental strategy

- Work incrementally!
- Start with a very simple model and get it to work!
  - It’s hard to fix a complex but broken model
- Add bells and whistles one-by-one and get the model working with each of them (or abandon them)

- Initially run on a tiny amount of data
  - You will see bugs much more easily on a tiny dataset … and they train really quickly
  - Something like 4–8 examples is good
  - Often synthetic data is useful for this
  - Make sure you can get 100% on this data (testing on train)
    - Otherwise your model is definitely either not powerful enough or it is broken

Experimental strategy

- Train and run your model on a large dataset
  - It should still score close to 100% on the training data after optimization
    - Otherwise, you probably want to consider a more powerful model!
    - Overfitting to training data is **not** something to fear when doing deep learning
      - These models are usually good at generalizing because of the way distributed representations share statistical strength regardless of overfitting to training data
- But, still, you now want good generalization performance:
  - Regularize your model until it doesn't overfit on dev data
    - Strategies like L2 regularization can be useful
    - But normally **generous dropout** is the secret to success

Details matter!

- Look at your data, collect summary statistics
- Look at your model's outputs, do error analysis
- Tuning hyperparameters, learning rates, getting initialization right, etc. is **often** important to the successes of NNets

Good luck with your projects!