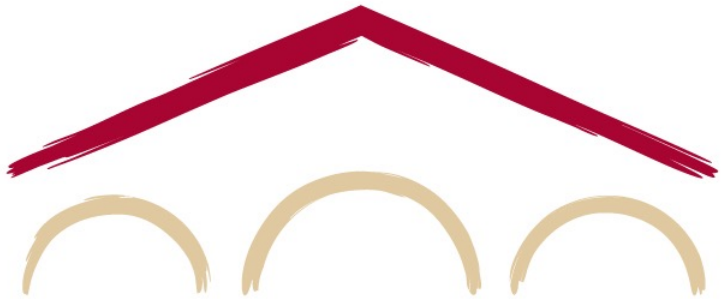
Natural Language Processing with Deep Learning CS224N/Ling284



Christopher Manning

Lecture 7: NMT and Final Projects; Practical Tips

Lecture Plan

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

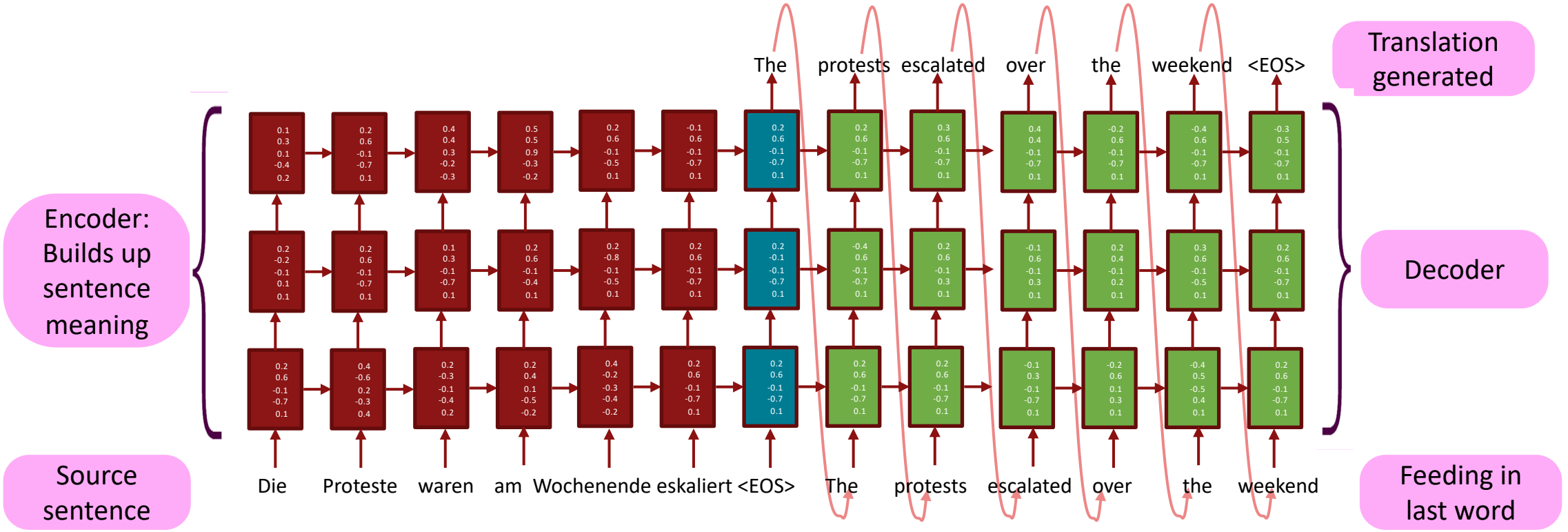
1. NMT [20 mins]
2. Attention [20 mins]
   – Mini Break –
3. Final projects types and details; assessment revisited [20 mins]
4. Finding research topics and sources of data [20 mins]

- Announcements
  - Assignment 3 is due today – I hope your dependency parsers are parsing text!
  - Assignment 4 out today – you get 9 days for it (!), due Thu
    - Get started early! It's bigger – and harder coding-wise – than the previous assignments 😰
    - **Starting with Ass 4, the TAs will no longer look at and debug your code for you!**

1. Multi-layer deep encoder-decoder machine translation net

[Sutskever et al. 2014; Luong et al. 2015]

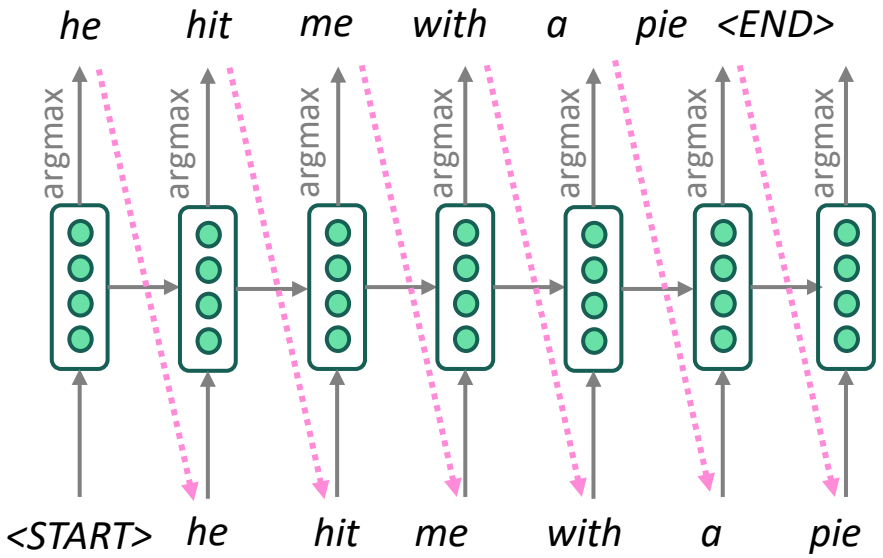
The hidden states from RNN layer *i* are the inputs to RNN layer *i* + 1



Conditioning = Bottleneck

Decoding: Greedy decoding

- We saw how to generate (or “decode”) the target sentence by taking argmax on each step of the decoder



- This is greedy decoding (take most probable word on each step)

Problems with greedy decoding

- Greedy decoding has no way to undo decisions!
  - Input: *il a m'entarté* *(he hit me with a pie)*
  - → *he* ____
  - → *he hit* ____
  - → *he hit a* ____ *(whoops! no going back now…)*

- How to fix this?

Exhaustive search decoding

- Ideally, we want to find a (length *T*) translation *y* that maximizes

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots,P(y_T|y_1,\ldots,y_{T-1},x)$$
$$= \prod_{t=1}^{T} P(y_t|y_1,\ldots,y_{t-1},x)$$

- We could try computing all possible sequences *y*
  - This means that on each step *t* of the decoder, we're tracking V^t possible partial translations, where *V* is vocab size
  - This $O(V^T)$ complexity is far too expensive!

Beam search decoding

- Core idea: On each step of decoder, keep track of the *k* most probable partial translations (which we call *hypotheses*)
  - *k* is the beam size (in practice around 5 to 10, in NMT)

- A hypothesis $y_1, \ldots, y_t$ has a score which is its log probability:

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

  - Scores are all negative, and higher score is better
  - We search for high-scoring hypotheses, tracking top *k* on each step

- Beam search is not guaranteed to find optimal solution
- But much more efficient than exhaustive search!

Beam search decoding: example

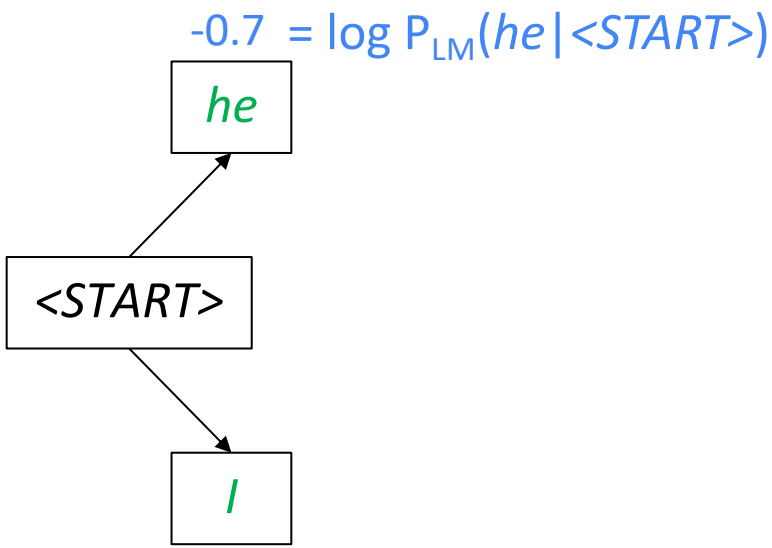
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

<START>

Calculate prob dist of next word

Beam search decoding: example

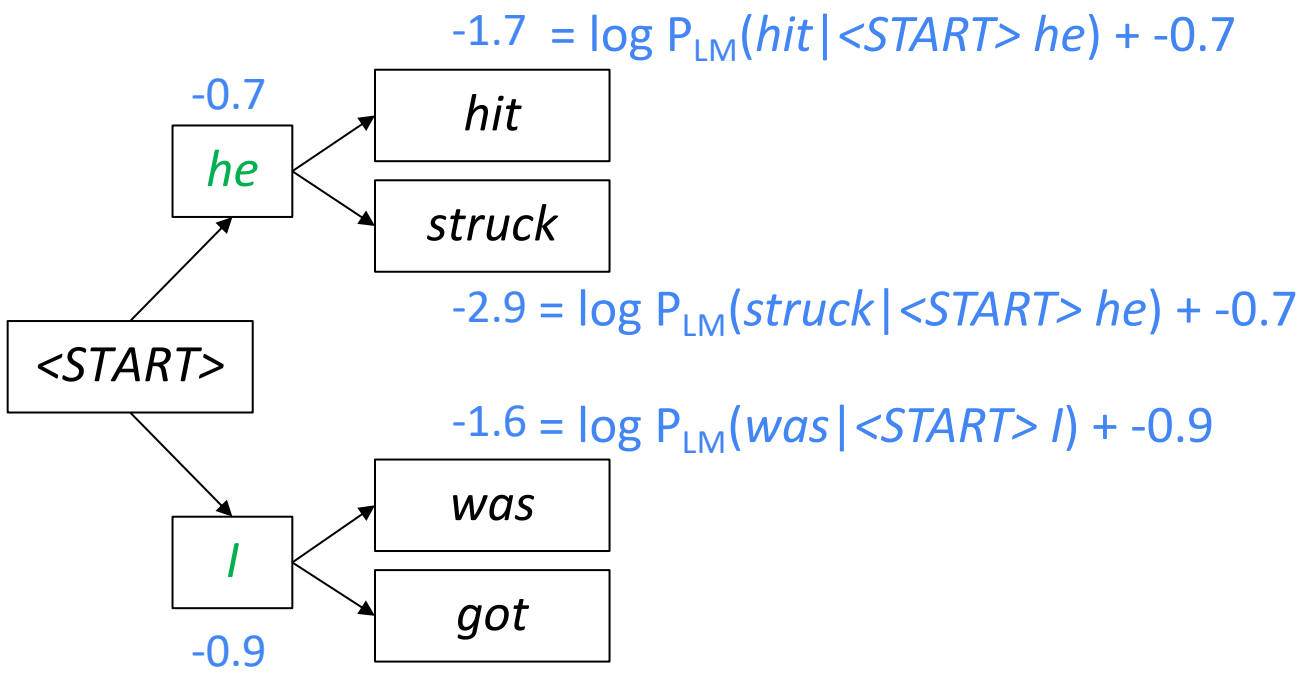
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Take top *k* words and compute scores

Beam search decoding: example

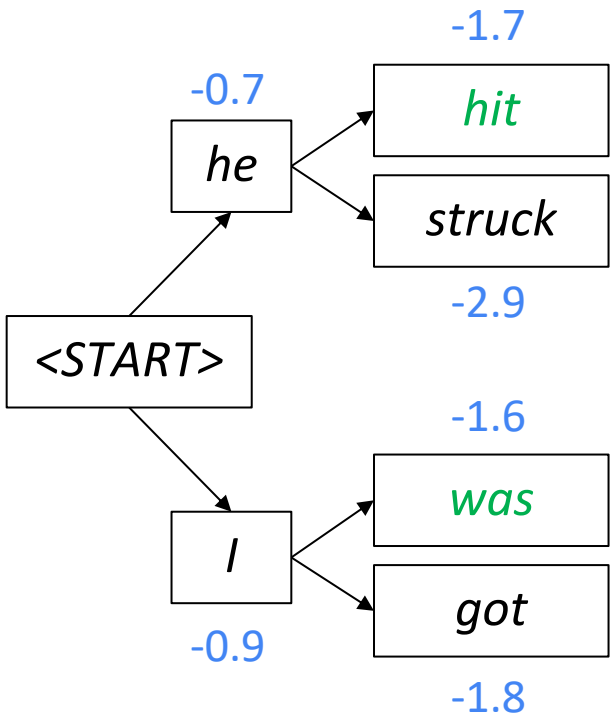
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



For each of the *k* hypotheses, find top *k* next words and calculate scores

Beam search decoding: example

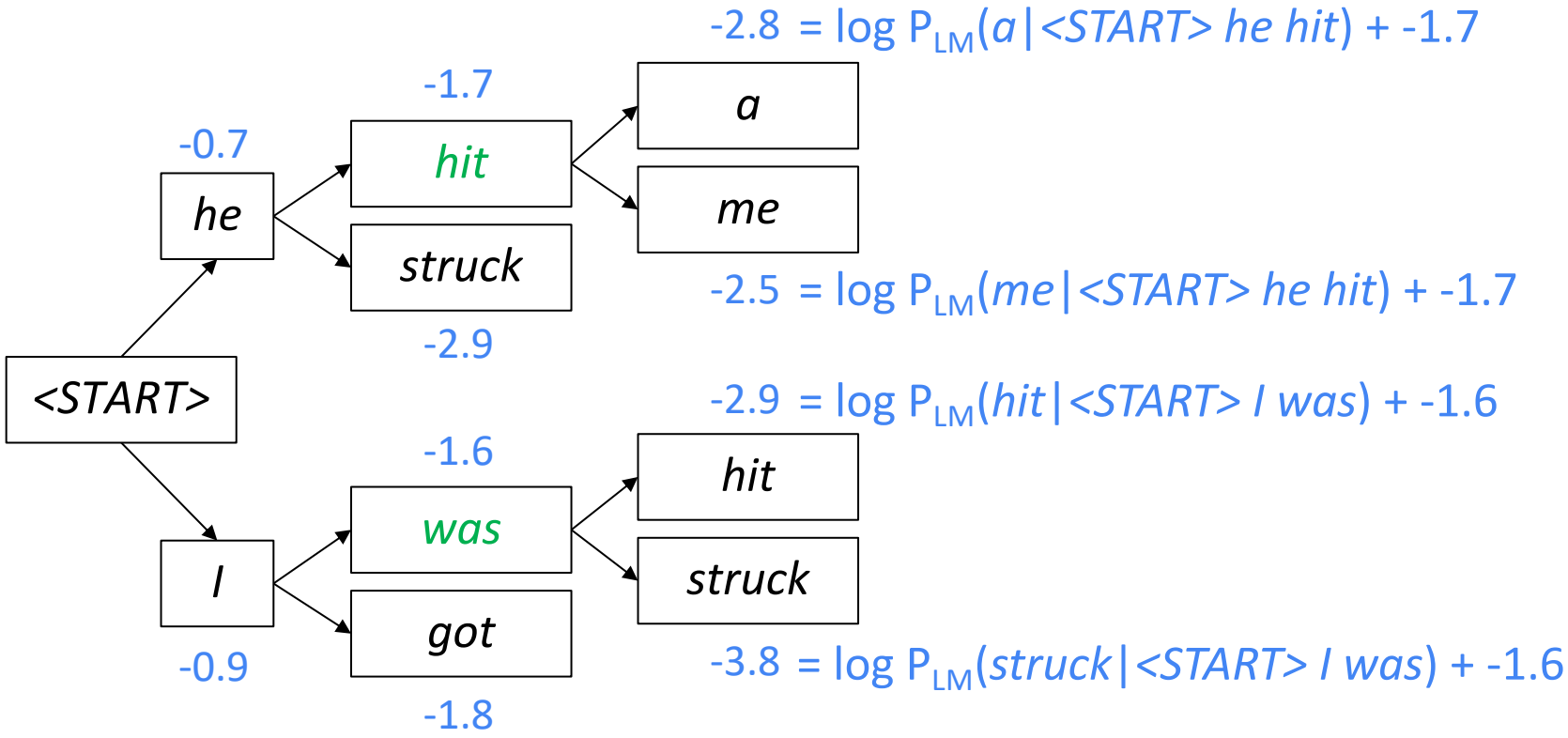
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Of these k^2 hypotheses,
just keep k with highest scores

Beam search decoding: example

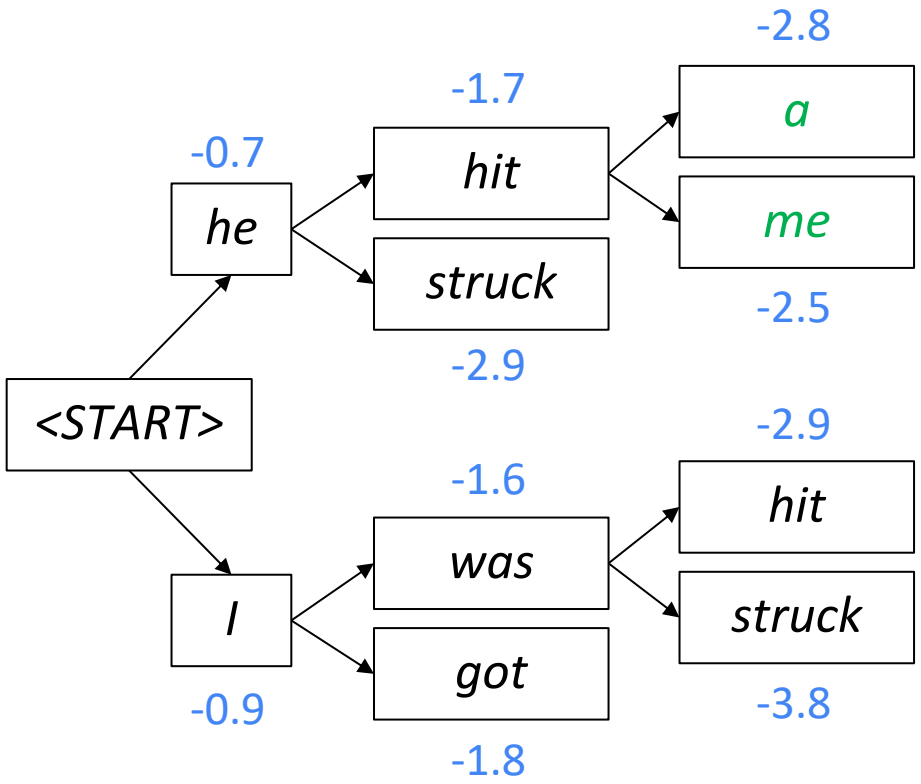
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



For each of the *k* hypotheses, find top *k* next words and calculate scores

Beam search decoding: example

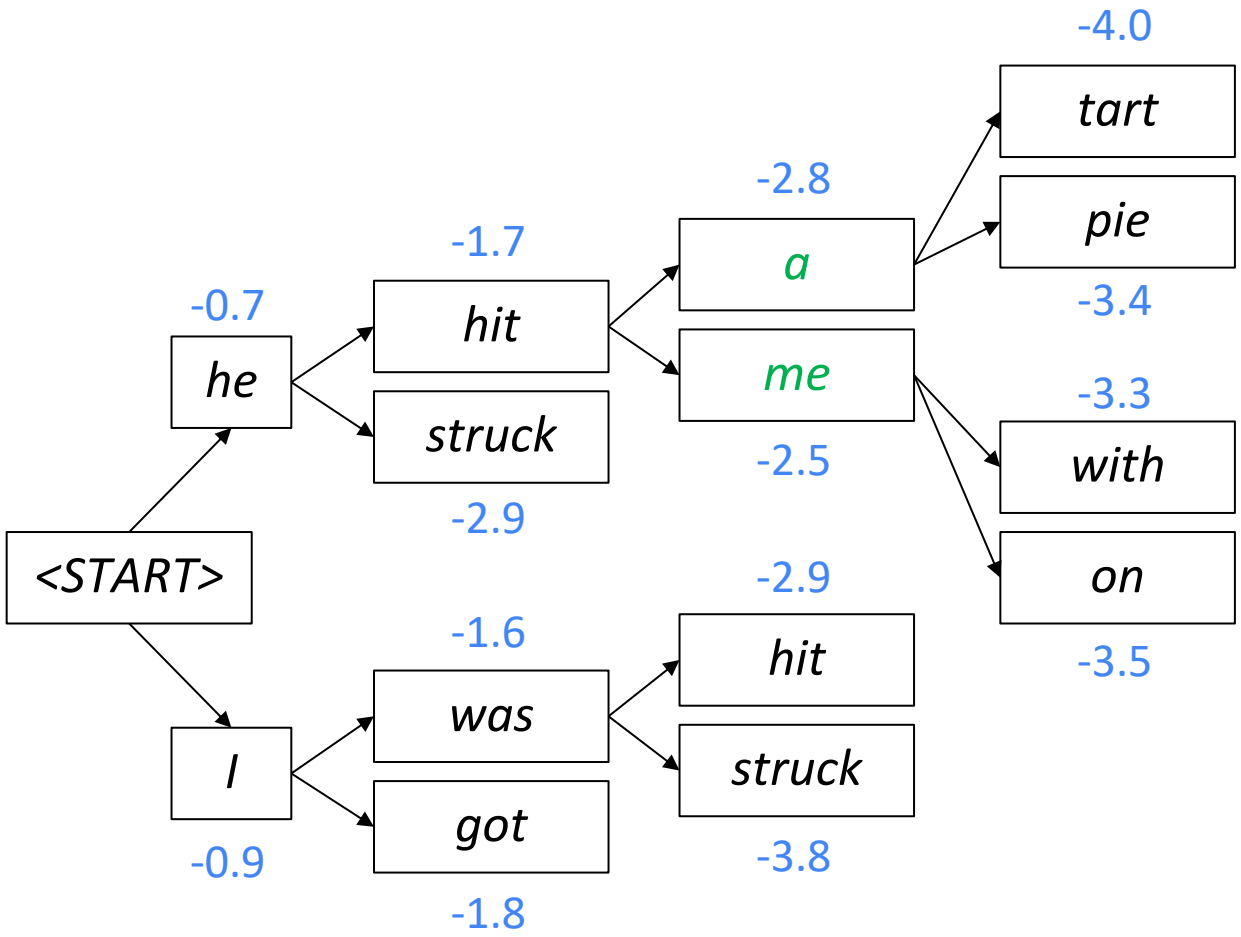
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Of these k^2 hypotheses,
just keep k with highest scores

Beam search decoding: example

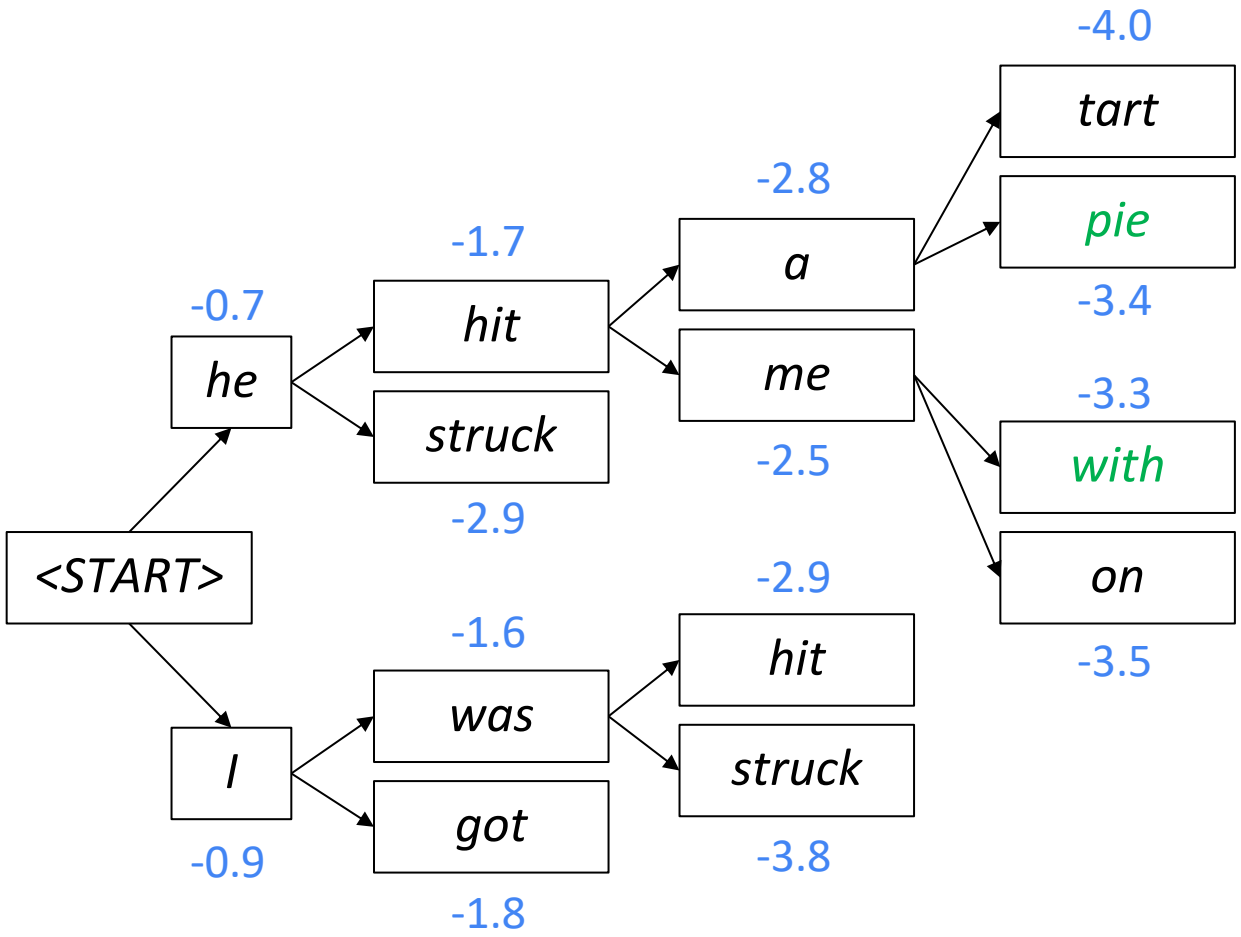
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



For each of the *k* hypotheses, find top *k* next words and calculate scores

Beam search decoding: example

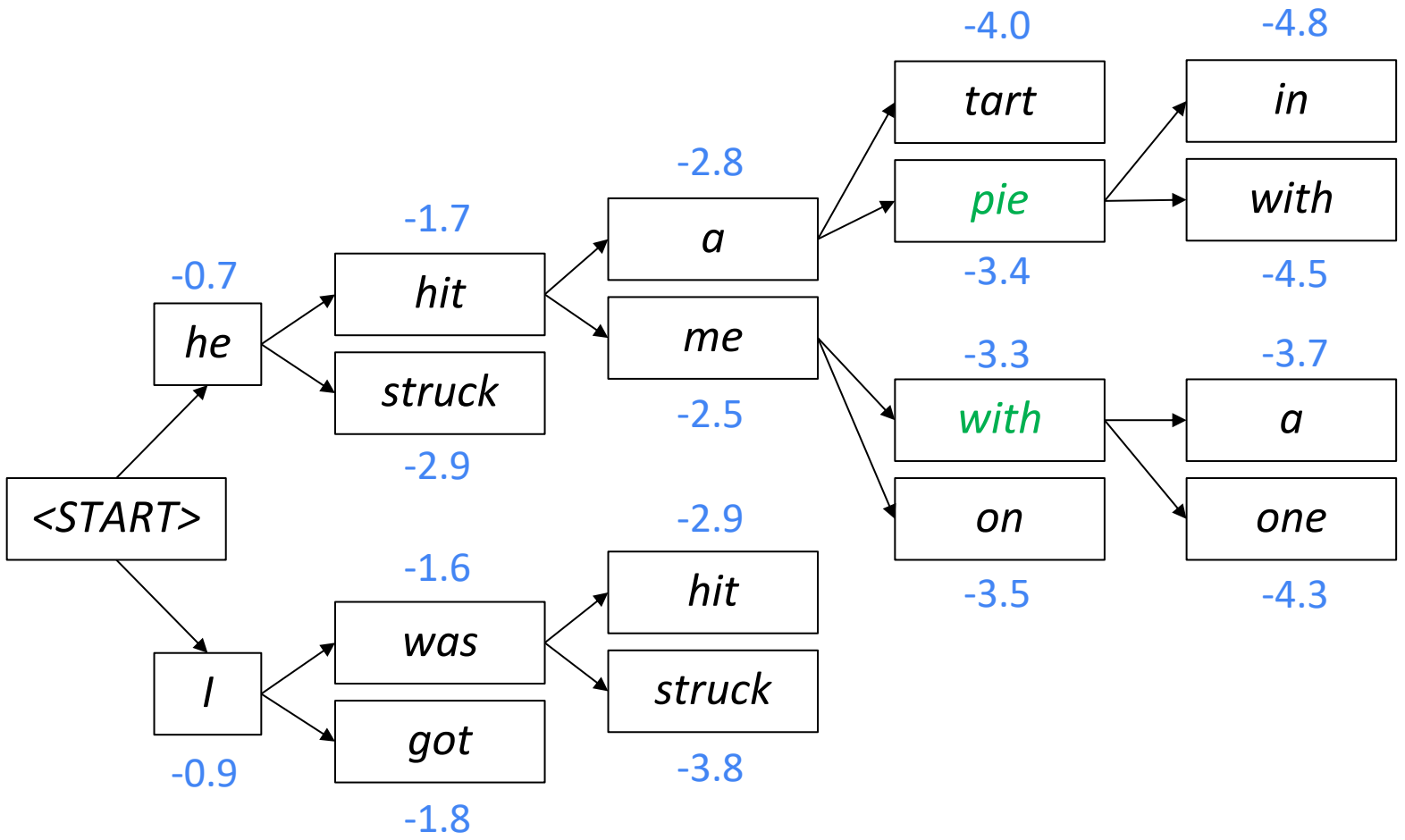
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Of these k^2 hypotheses,
just keep k with highest scores

Beam search decoding: example

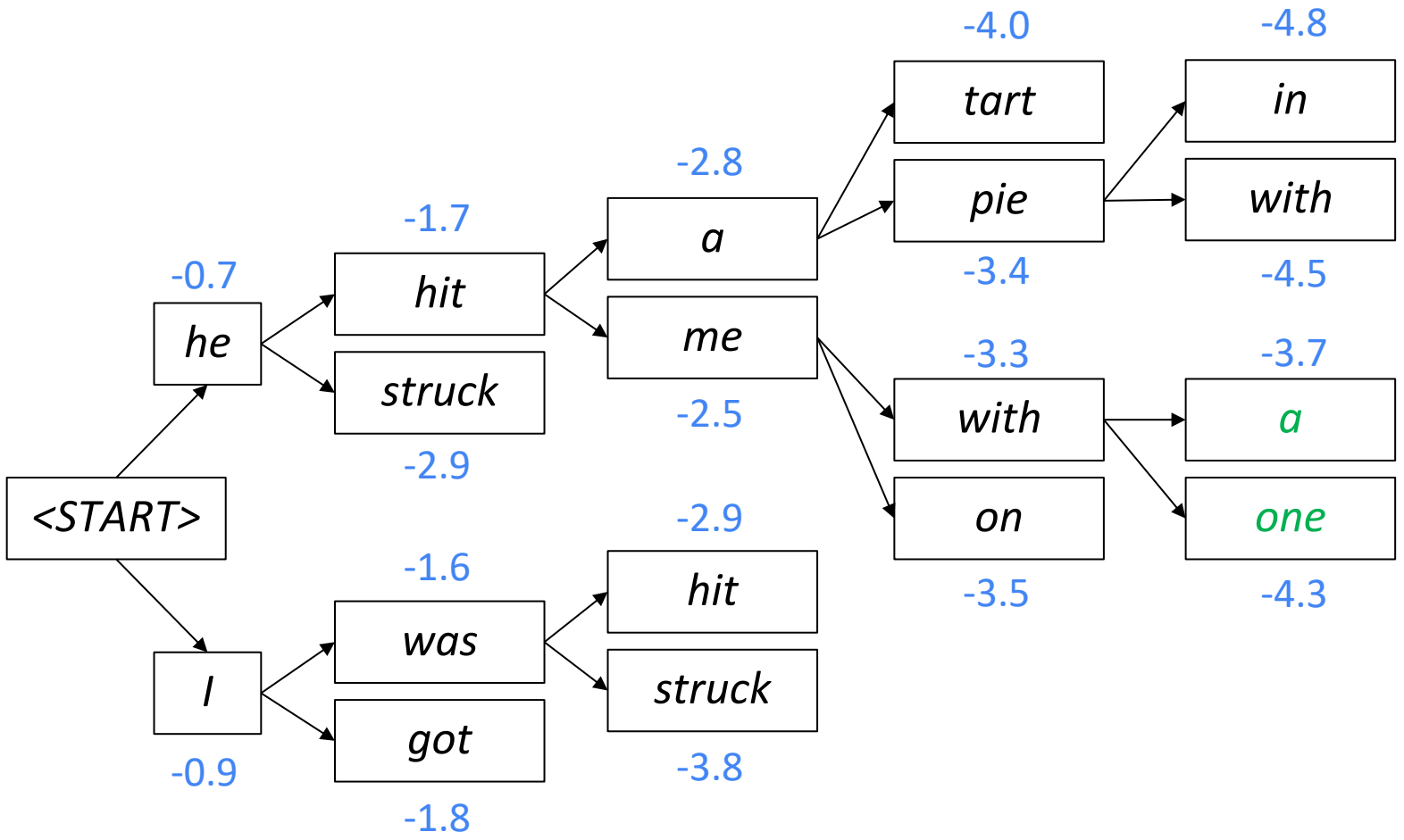
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



For each of the *k* hypotheses, find top *k* next words and calculate scores

Beam search decoding: example

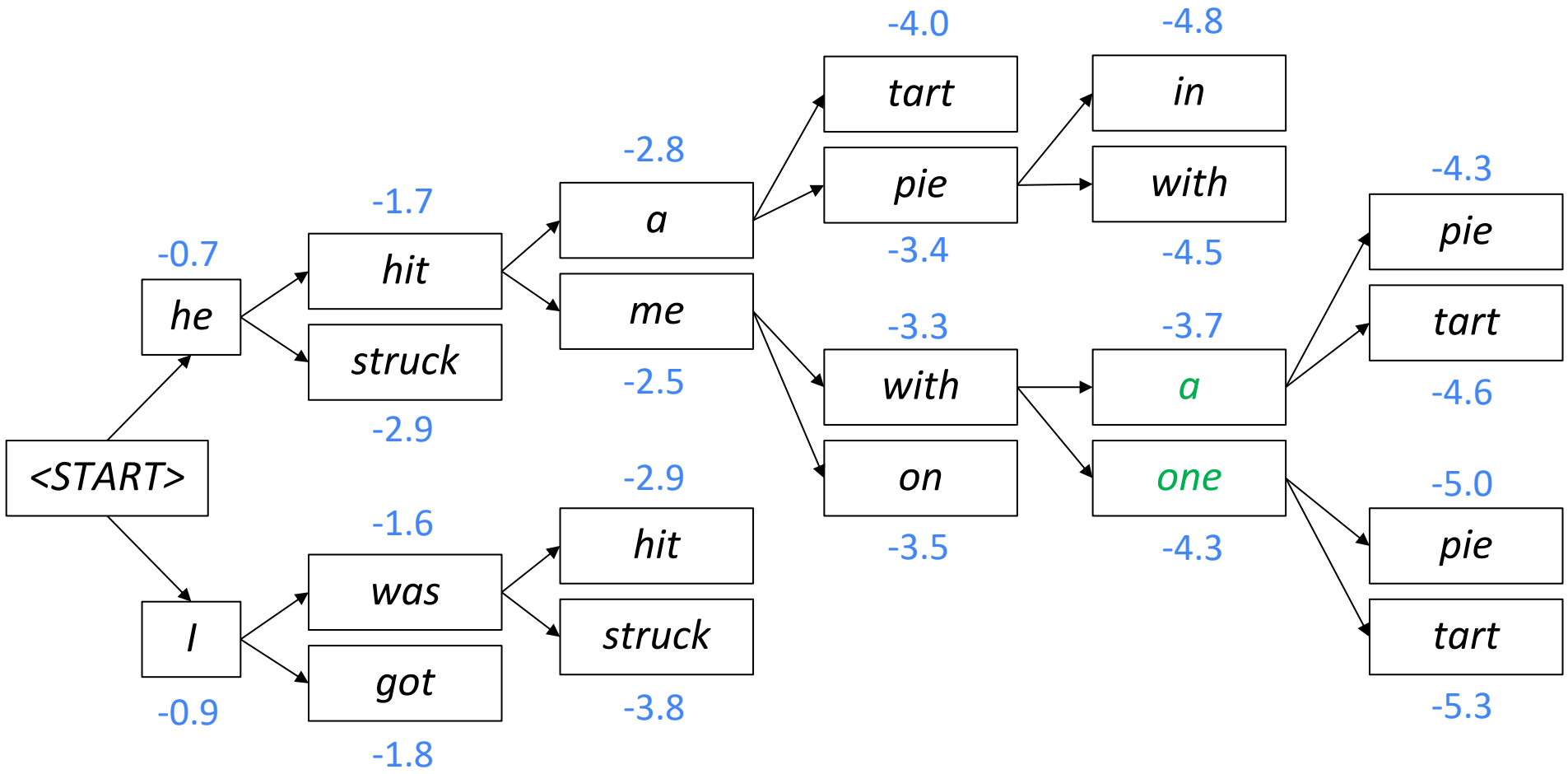
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Of these k^2 hypotheses, just keep k with highest scores

Beam search decoding: example

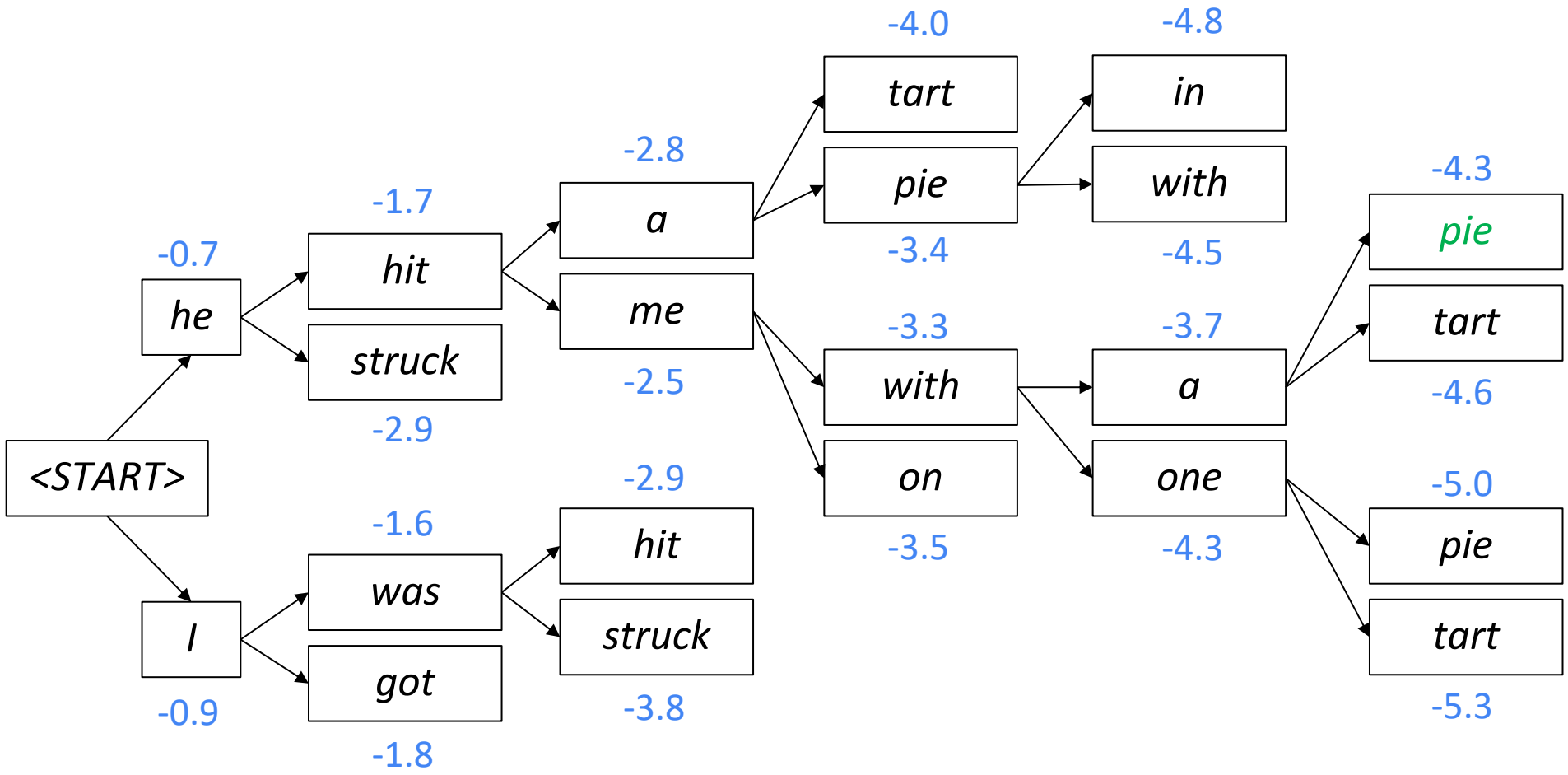
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



For each of the *k* hypotheses, find top *k* next words and calculate scores

Beam search decoding: example

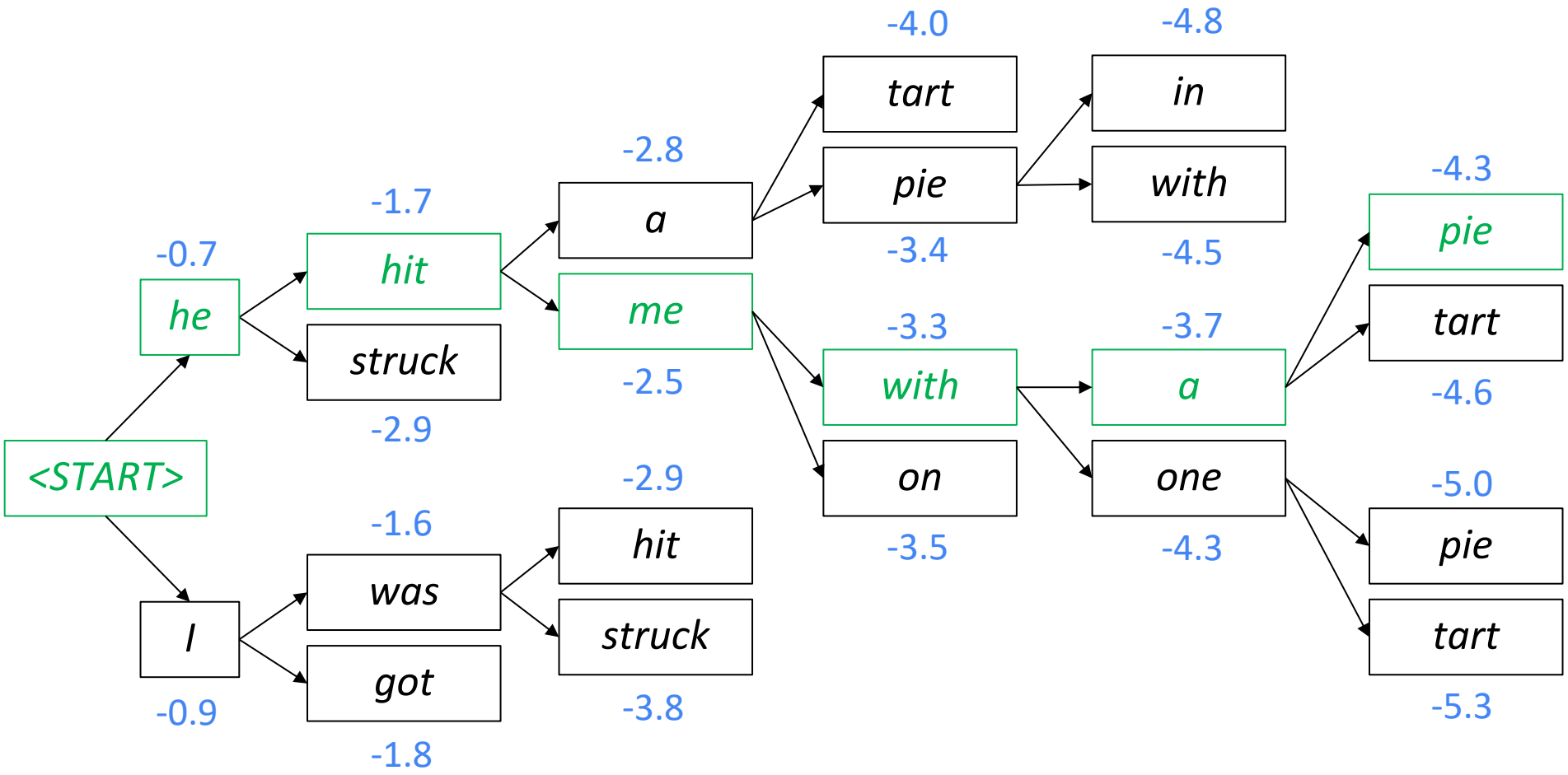
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



This is the top-scoring hypothesis!

Beam search decoding: example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Backtrack to obtain the full hypothesis

Beam search decoding: stopping criterion

- In greedy decoding, usually we decode until the model produces an <END> token
  - **For example:** *<START> he hit me with a pie <END>*

- In beam search decoding, different hypotheses may produce <END> tokens on different timesteps
  - When a hypothesis produces <END>, that hypothesis is complete.
  - Place it aside and continue exploring other hypotheses via beam search.

- Usually we continue beam search until:
  - We reach timestep *T* (where *T* is some pre-defined cutoff), or
  - We have at least *n* completed hypotheses (where *n* is pre-defined cutoff)

Beam search decoding: finishing up

- We have our list of completed hypotheses.
- How to select top one?

- Each hypothesis $y_1, \ldots, y_t$ on our list has a score

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

- **Problem with this:** longer hypotheses have lower scores

- **Fix:** Normalize by length. Use this to select top one instead:

$$\frac{1}{t} \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

See also discussion of sampling-based decoding in the NLG lecture

How do we evaluate Machine Translation?

BLEU (**Bi**lingual **E**valuation **U**nderstudy)

You'll see BLEU in detail in Assignment 4!

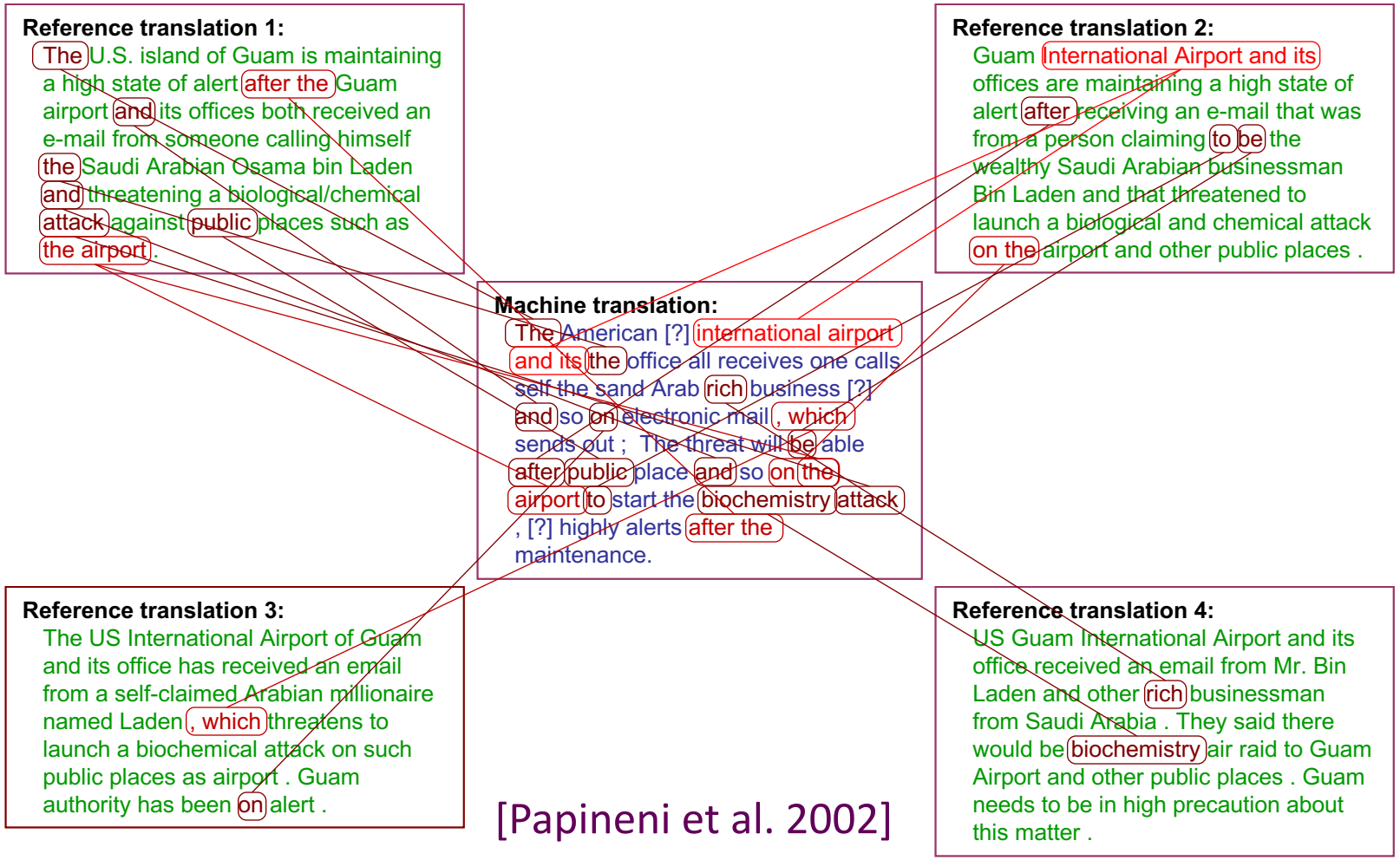
- BLEU compares the machine-written translation to one or several human-written translation(s), and computes a similarity score based on:
  - Geometric mean of *n*-gram precision (usually for 1, 2, 3 and 4-grams)
  - Plus a penalty for too-short system translations

- BLEU is useful but imperfect
  - There are many valid ways to translate a sentence
  - So a good translation can get a poor BLEU score because it has low *n*-gram overlap with the human translation ☹

See discussion of evaluation in NLG lecture

Source: "BLEU: a Method for Automatic Evaluation of Machine Translation", Papineni et al, 2002. http://aclweb.org/anthology/P02-1040

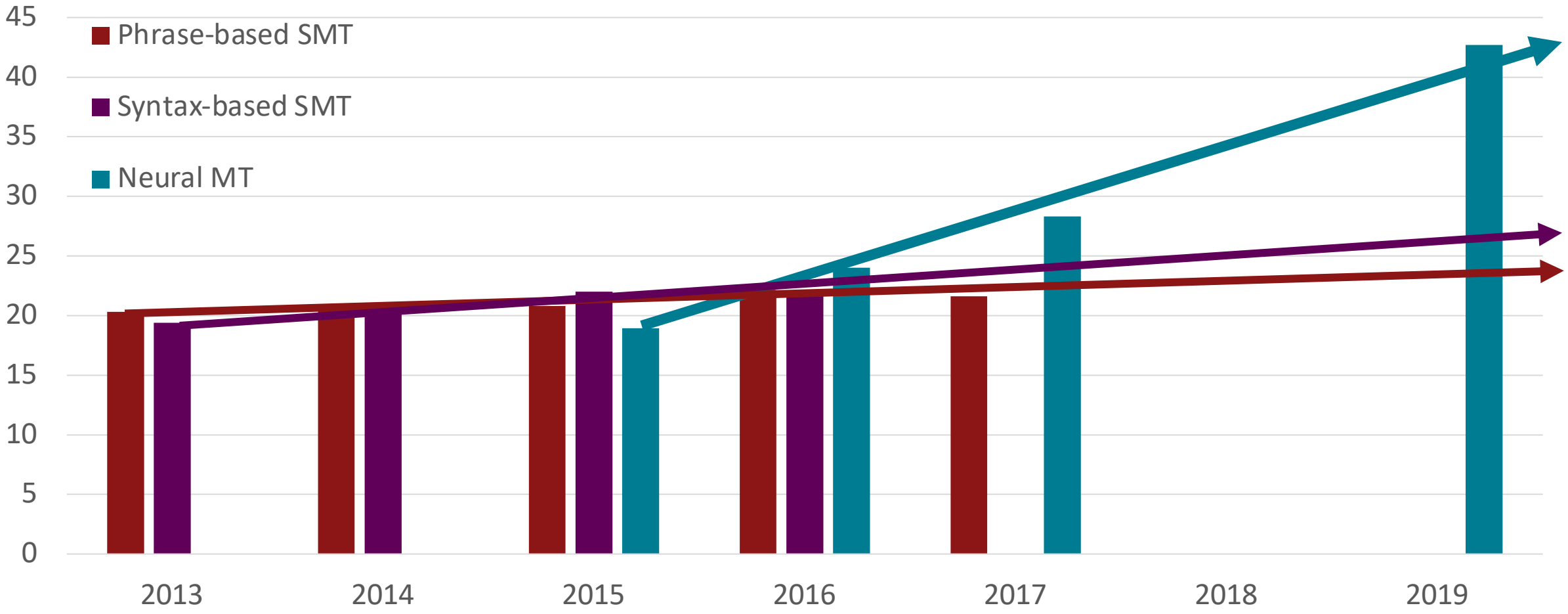
BLEU score against 4 reference translations



[Papineni et al. 2002]

MT progress over time

[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal; NMT 2019 FAIR on newstest2019]



Sources: http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf & http://matrix.statmt.org/

Advantages of NMT

Compared to SMT, NMT has many advantages:

- Better performance
  - More fluent
  - Better use of context
  - Better use of phrase similarities
- A single neural network to be optimized end-to-end
  - No subcomponents to be individually optimized
- Requires much less human engineering effort
  - No feature engineering
  - Same method for all language pairs

Disadvantages of NMT?

Compared to SMT:

- NMT is less interpretable
  - Hard to debug

- NMT is difficult to control
  - For example, can't easily specify rules or guidelines for translation
  - Safety concerns!
    - Invention of content not in source
    - Systematic gender biases

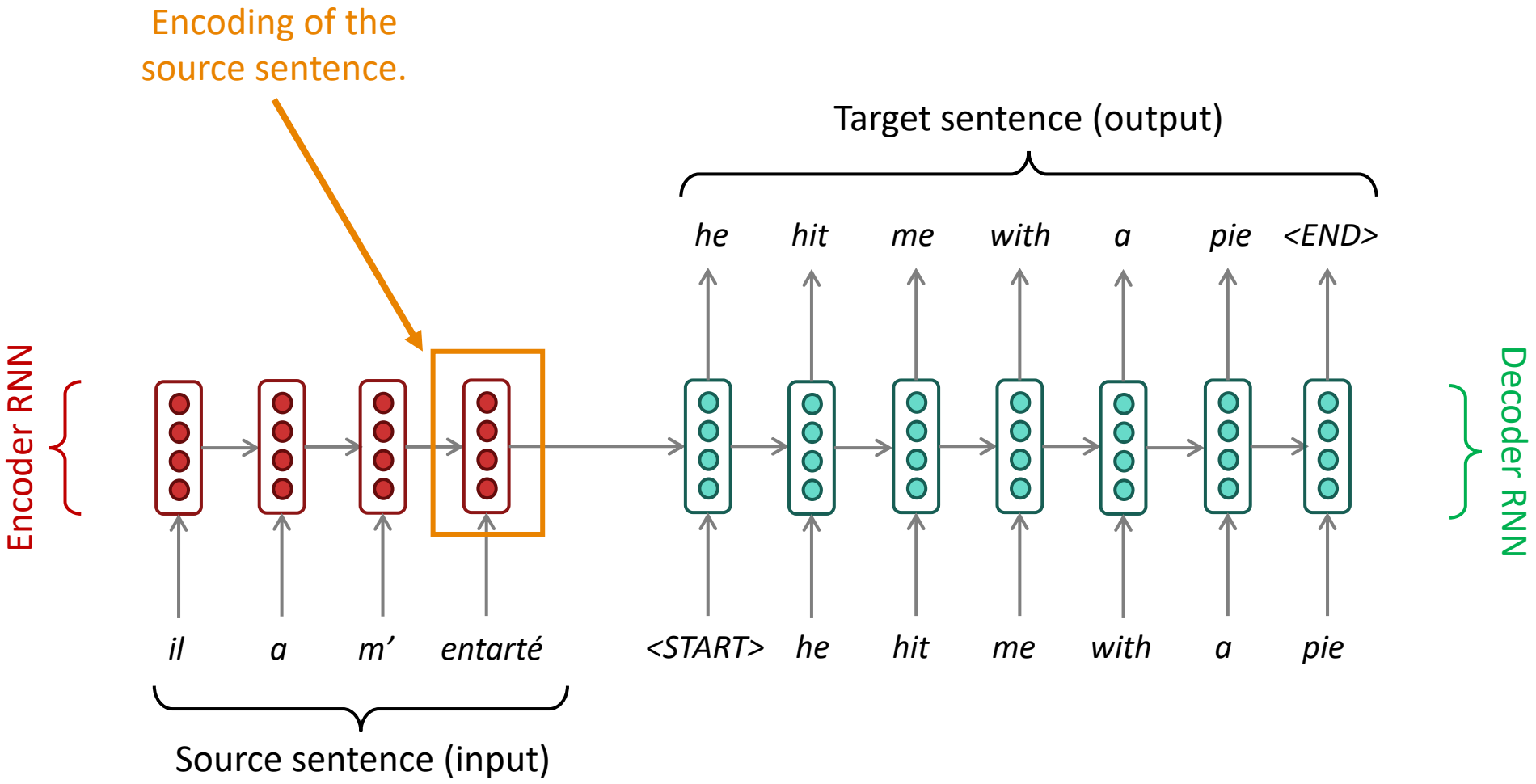
NMT: the first big success story of NLP Deep Learning

Neural Machine Translation went from a fringe research attempt in **2014** to the leading standard method in **2016**

- **2014**: First seq2seq paper published [Sutskever et al. 2014]
- **2016**: Google Translate switches from SMT to NMT – and by 2018 everyone has

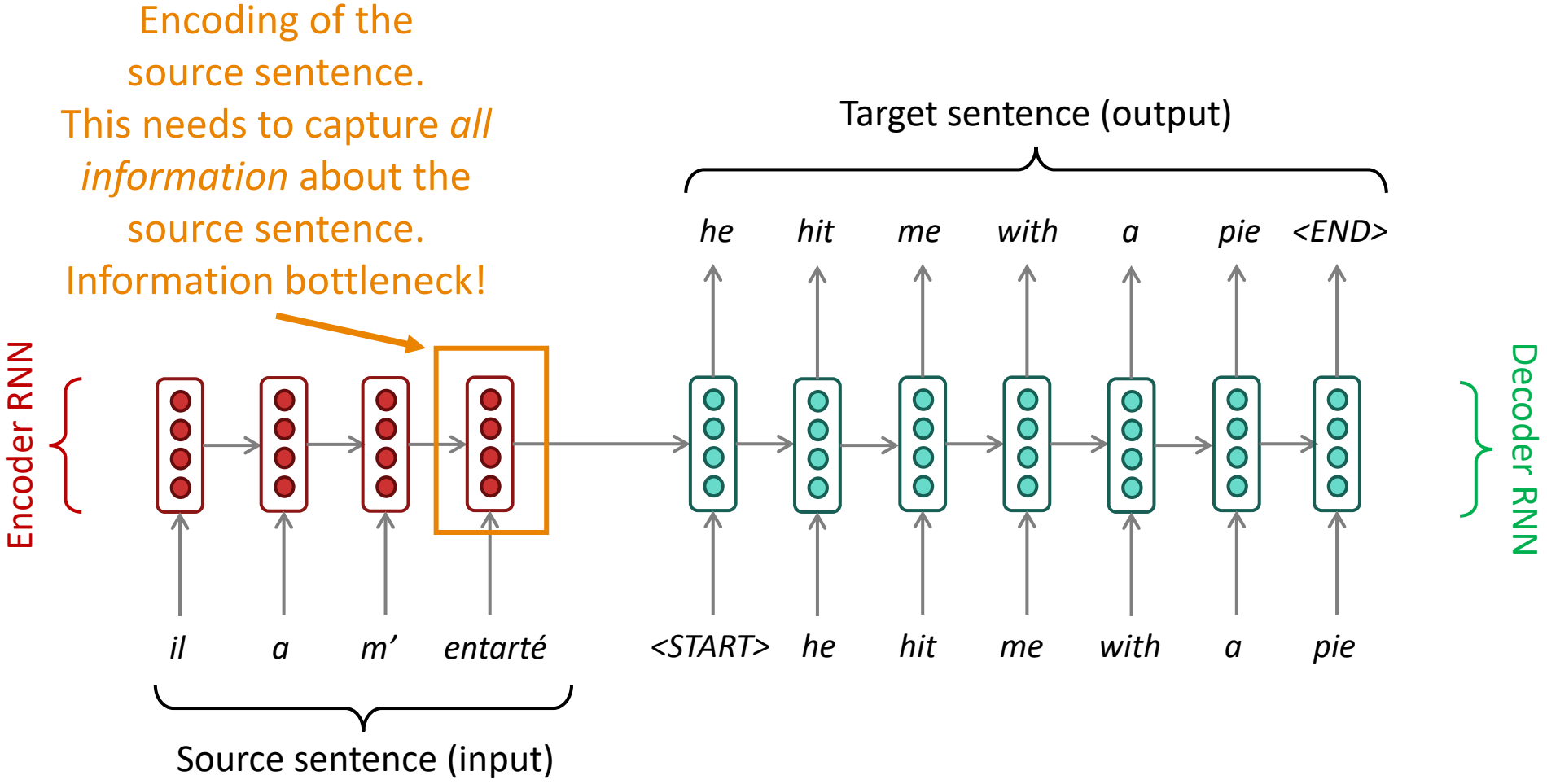
- This is amazing!
  - **SMT** systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by small groups of engineers in a few months

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



Problems with this architecture?

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



Attention

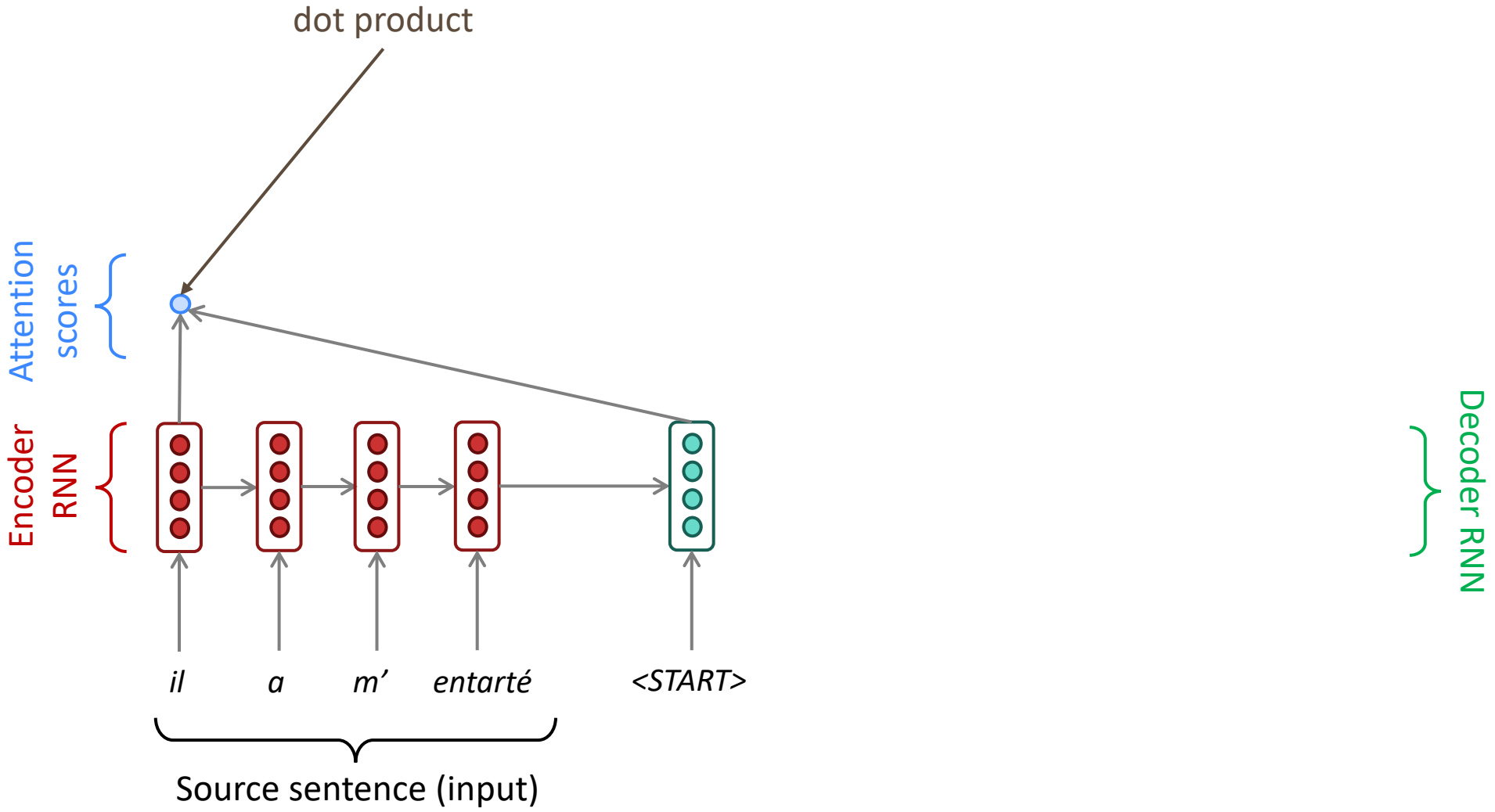
- **Attention** provides a solution to the bottleneck problem.

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

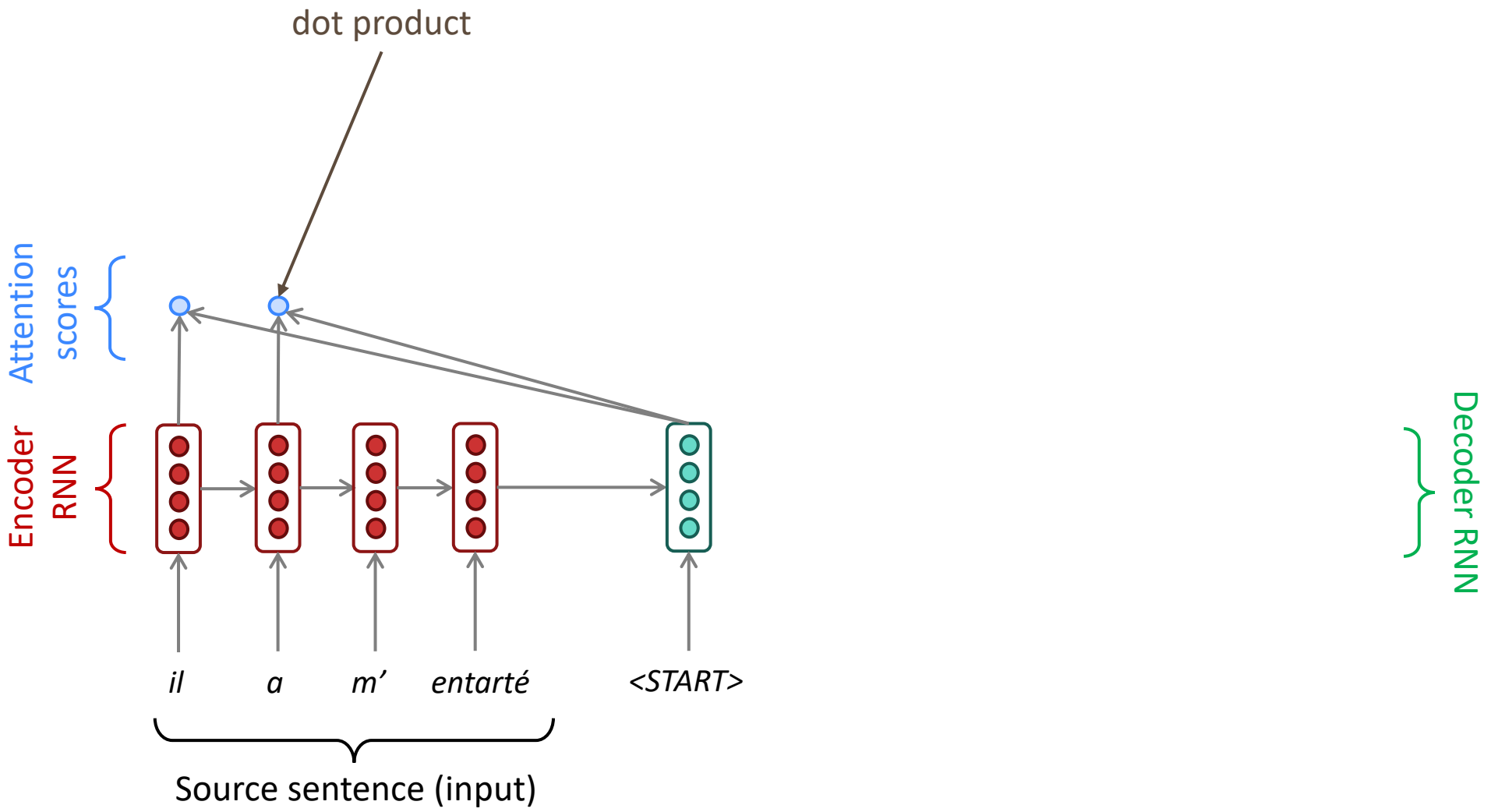
- First, we will show via diagram (no equations), then we will show with equations

Sequence-to-sequence with attention

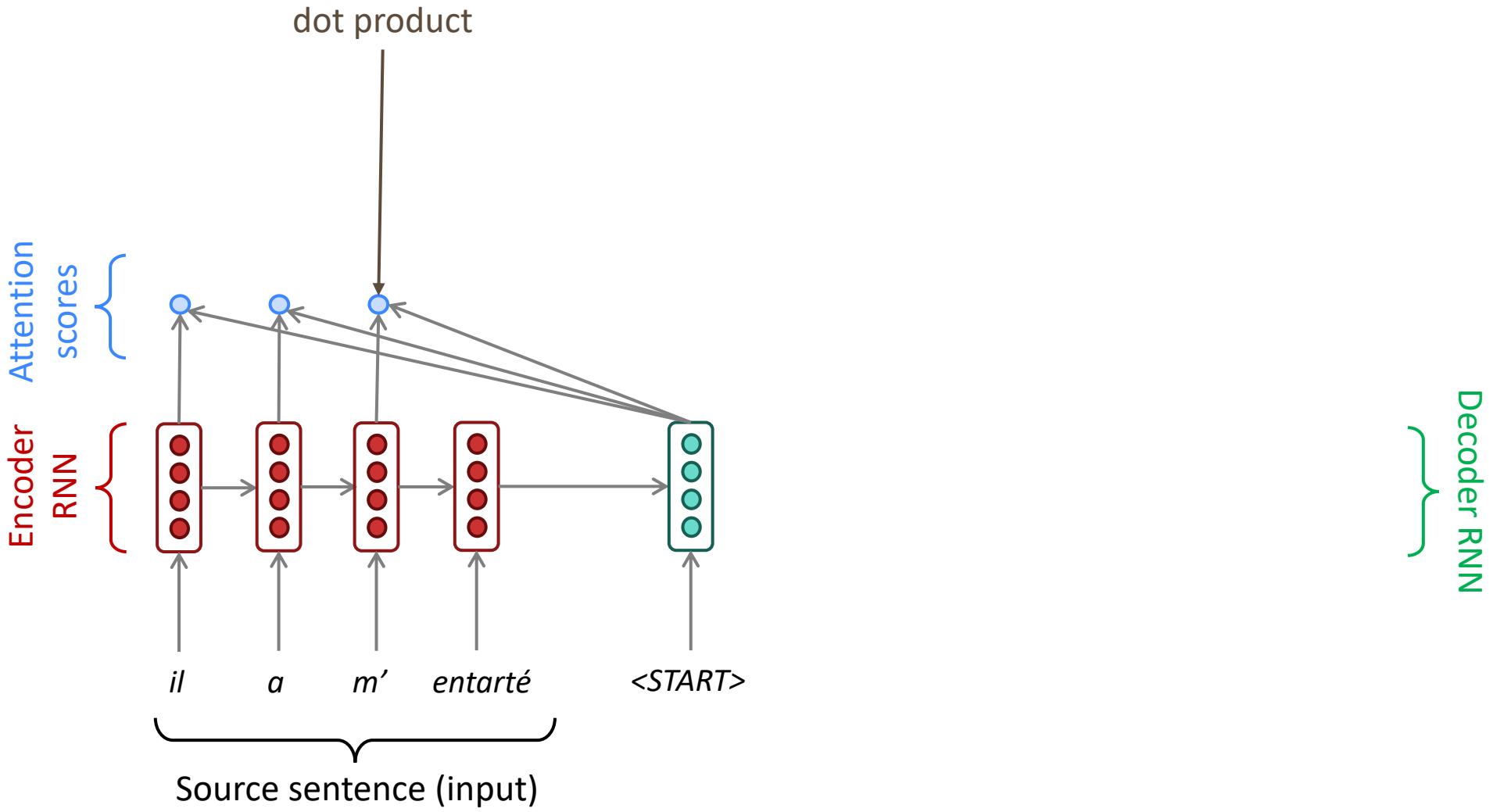
Core idea: on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence



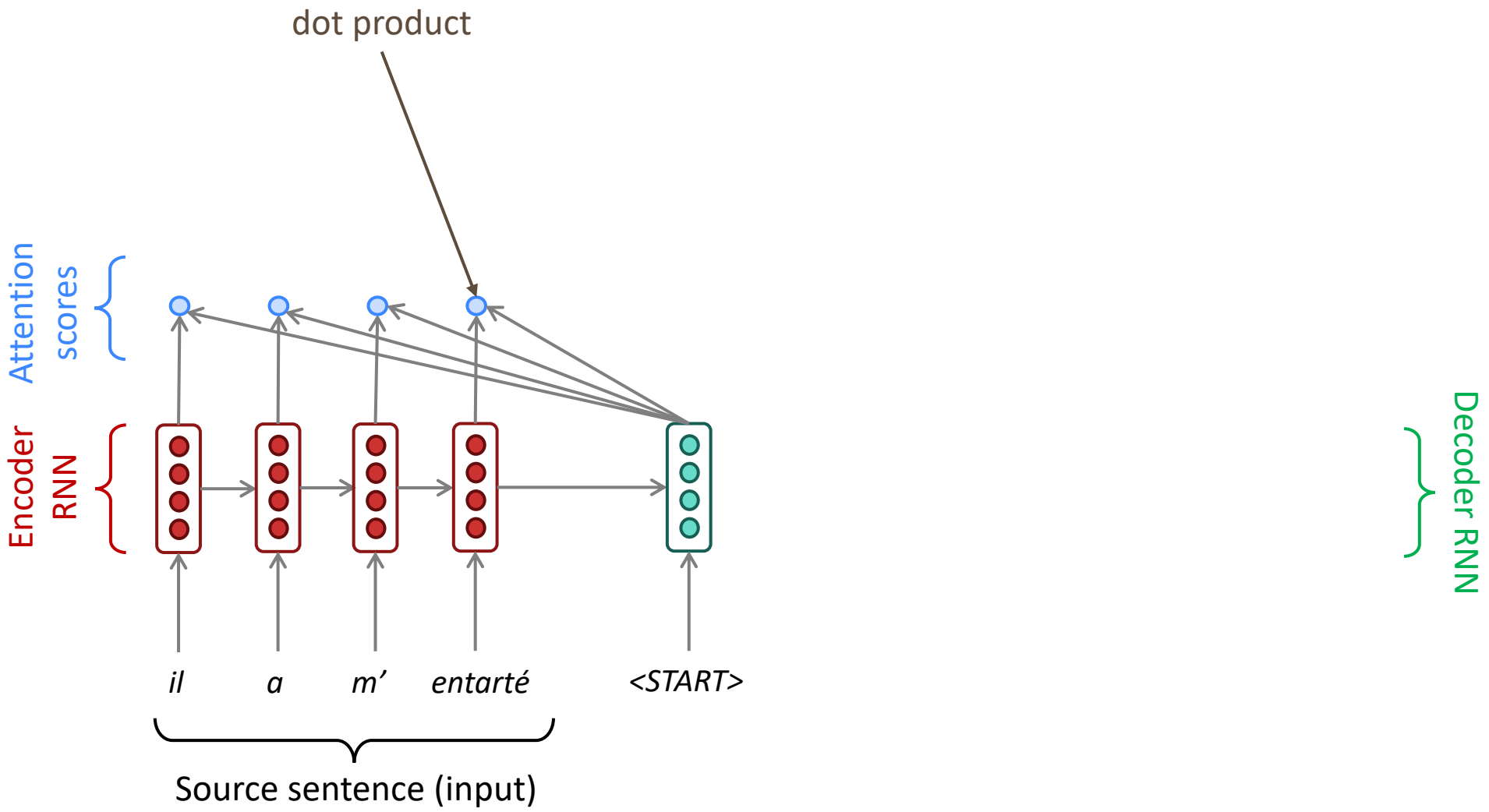
Sequence-to-sequence with attention



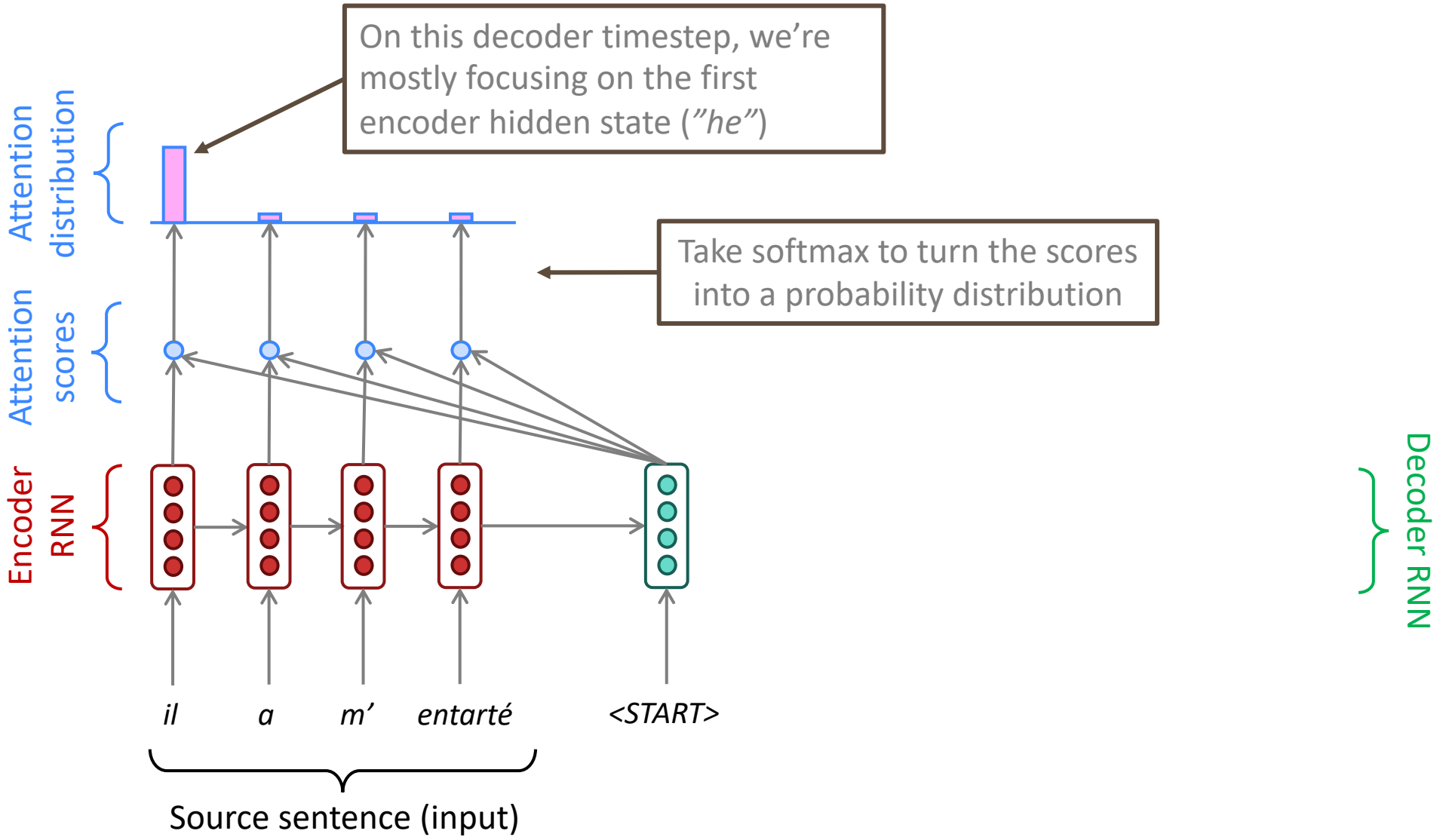
Sequence-to-sequence with attention



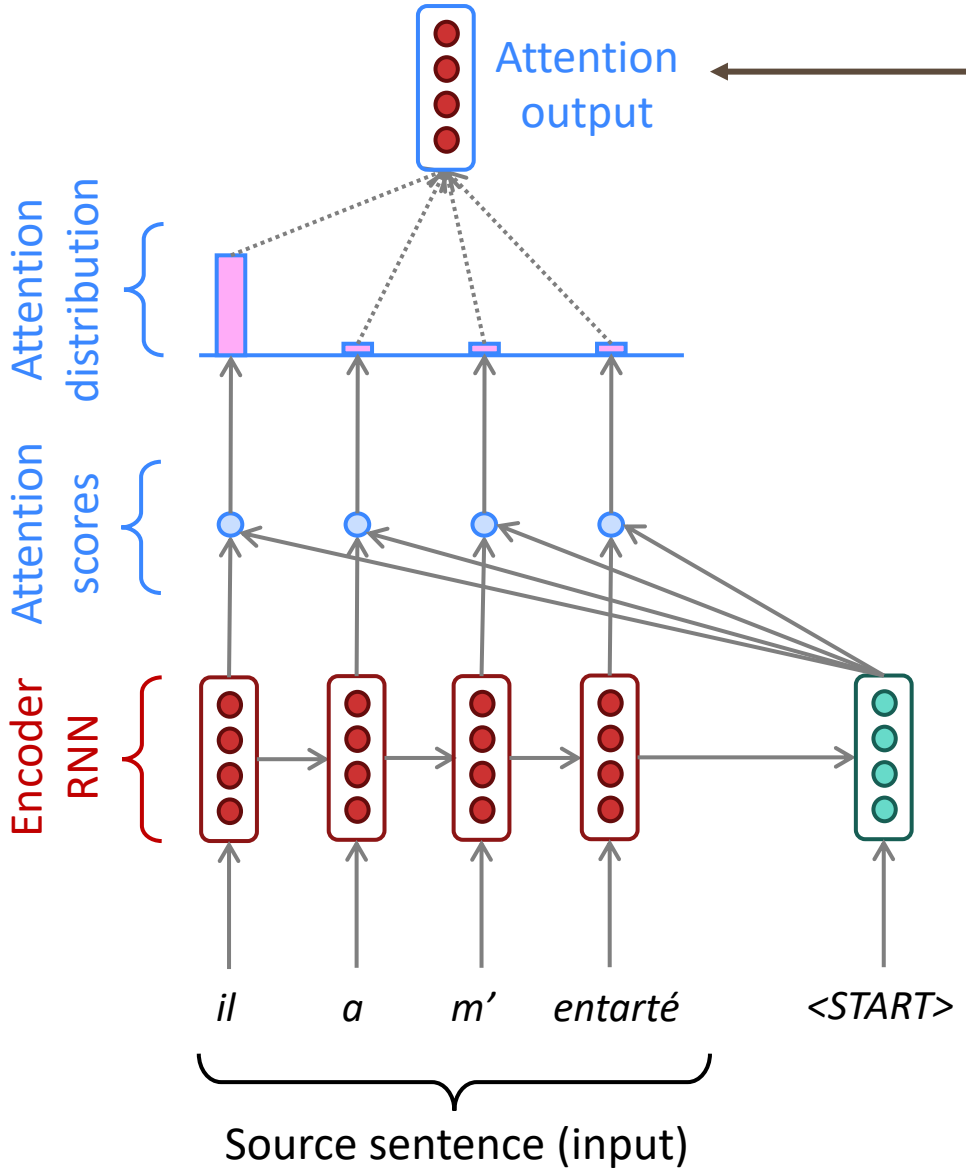
Sequence-to-sequence with attention



Sequence-to-sequence with attention



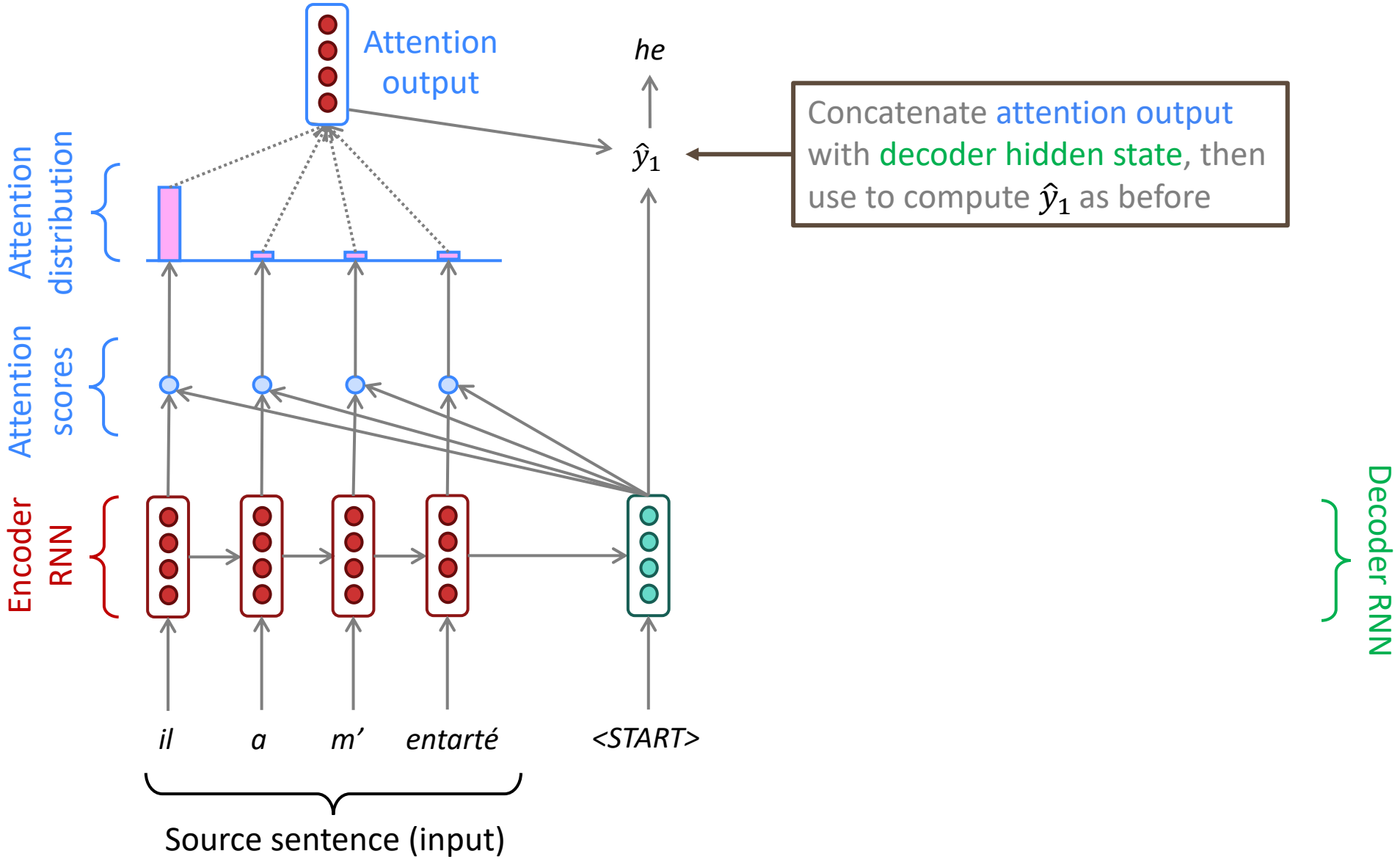
Sequence-to-sequence with attention



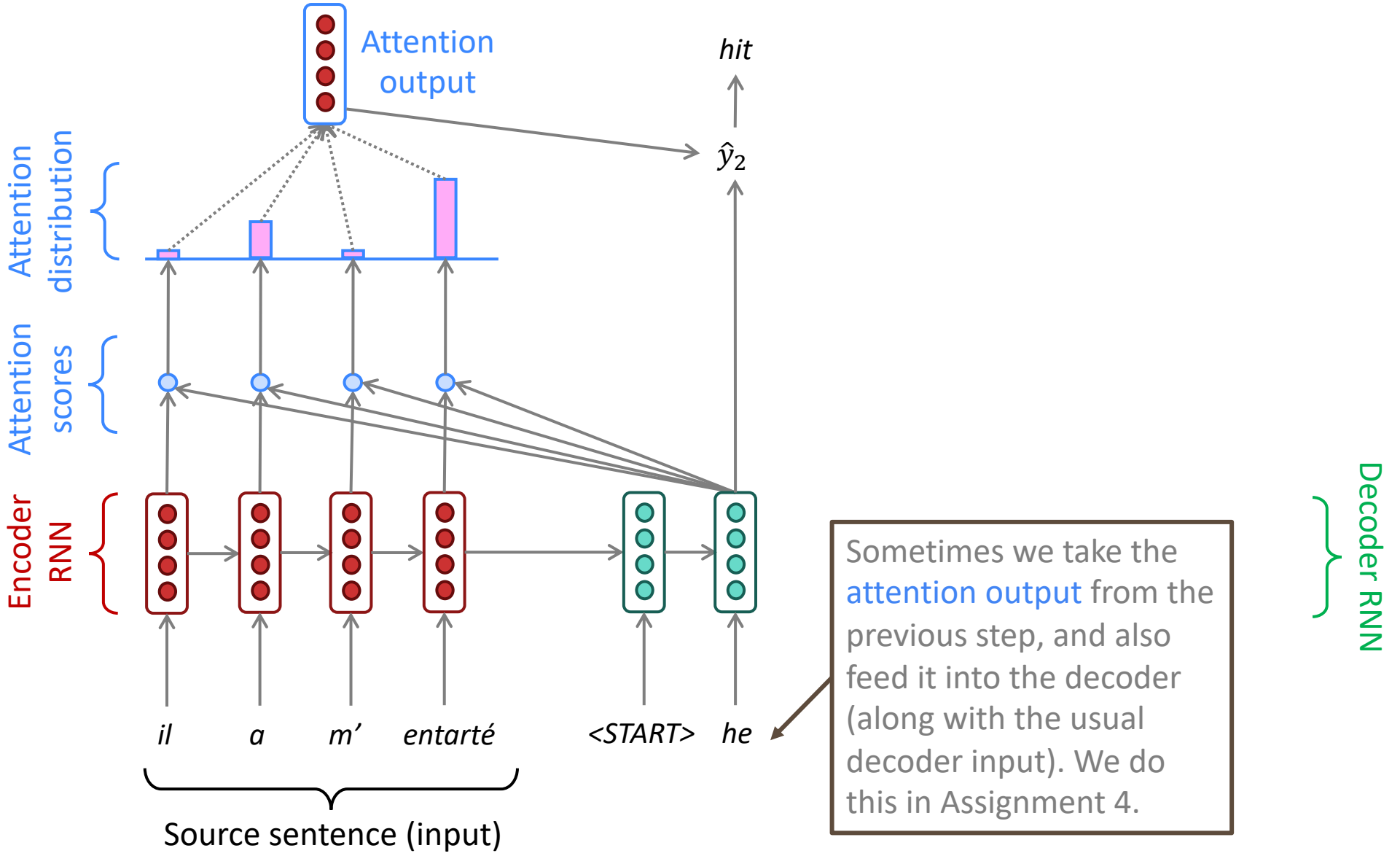
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

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

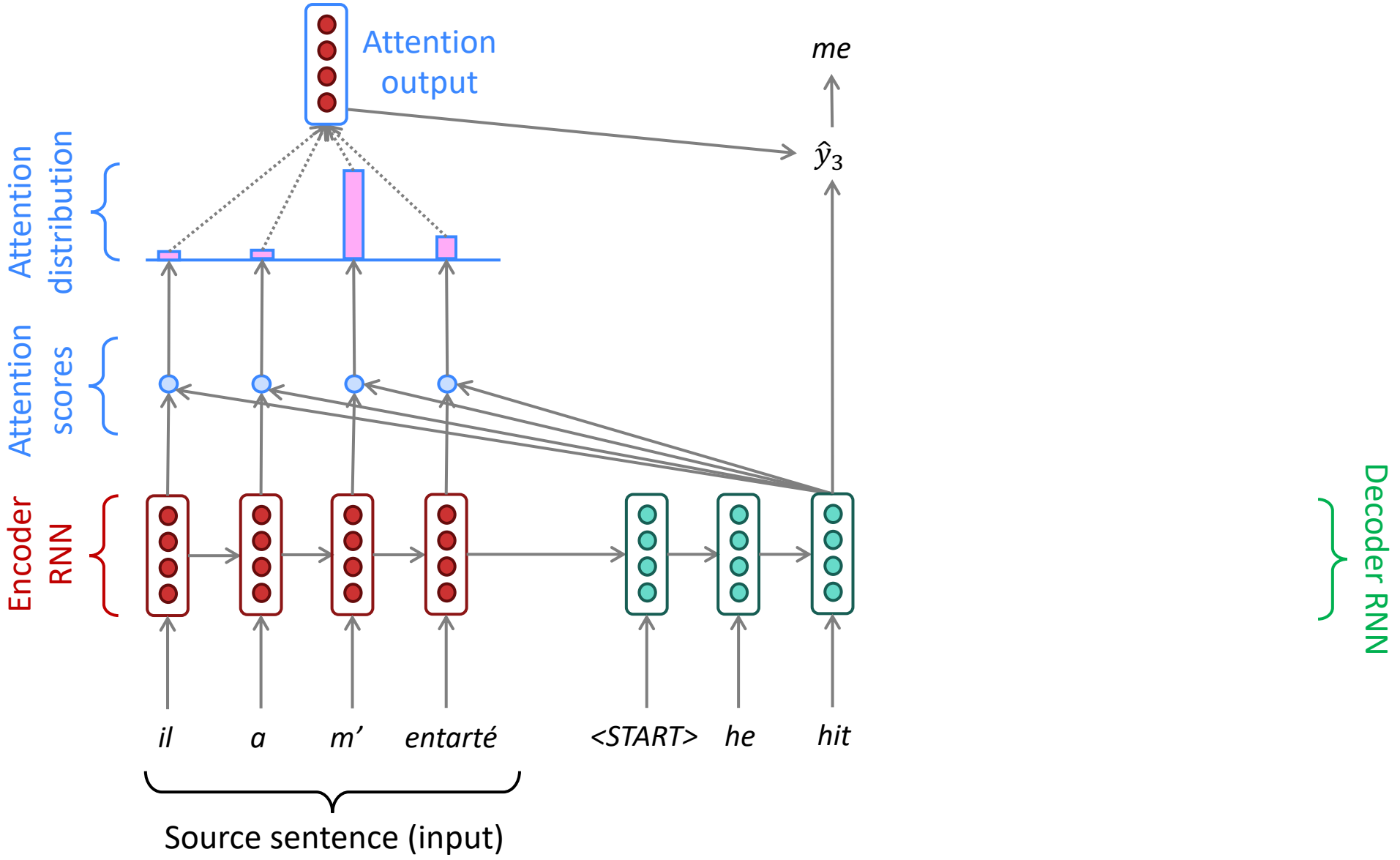
Sequence-to-sequence with attention



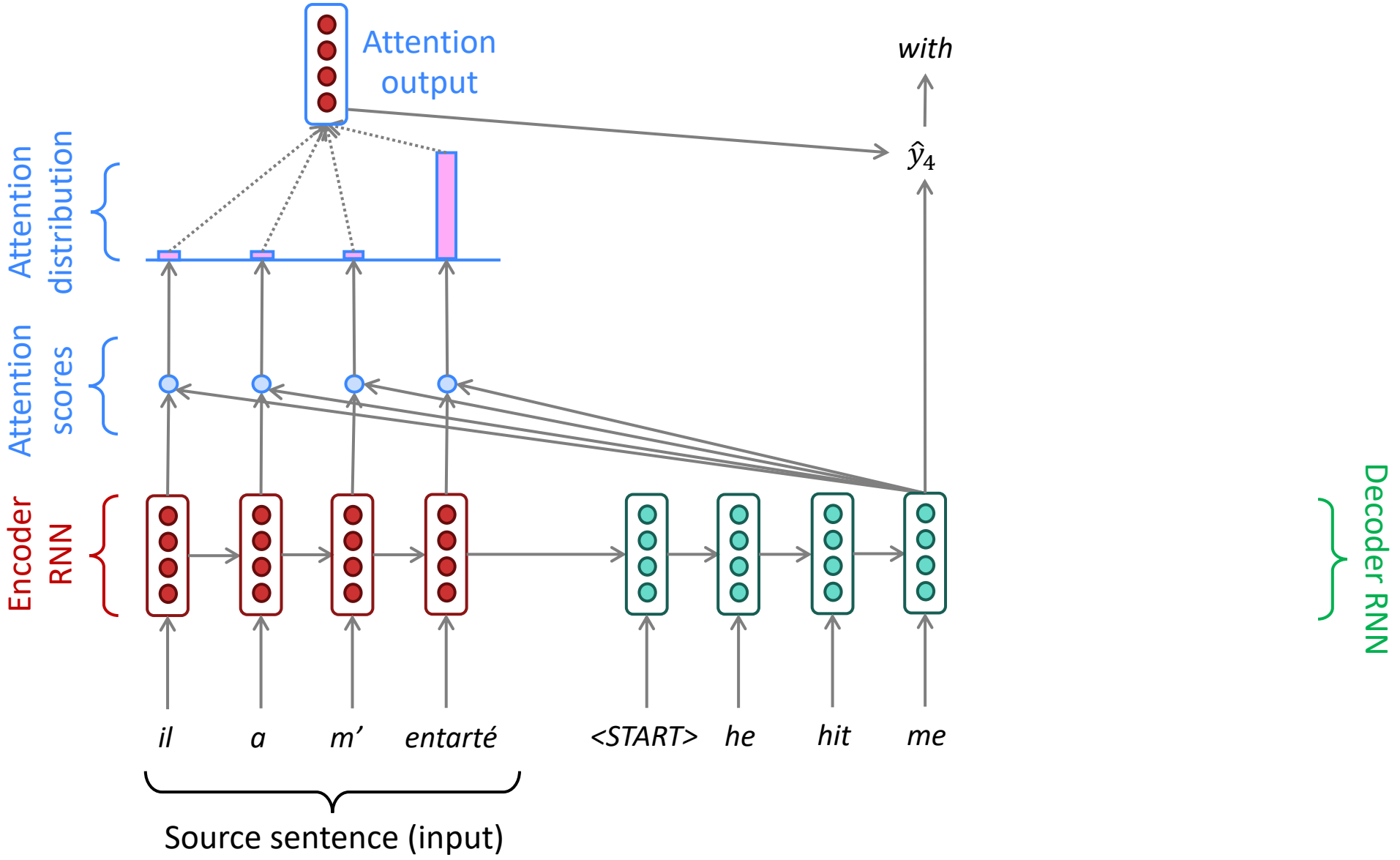
Sequence-to-sequence with attention



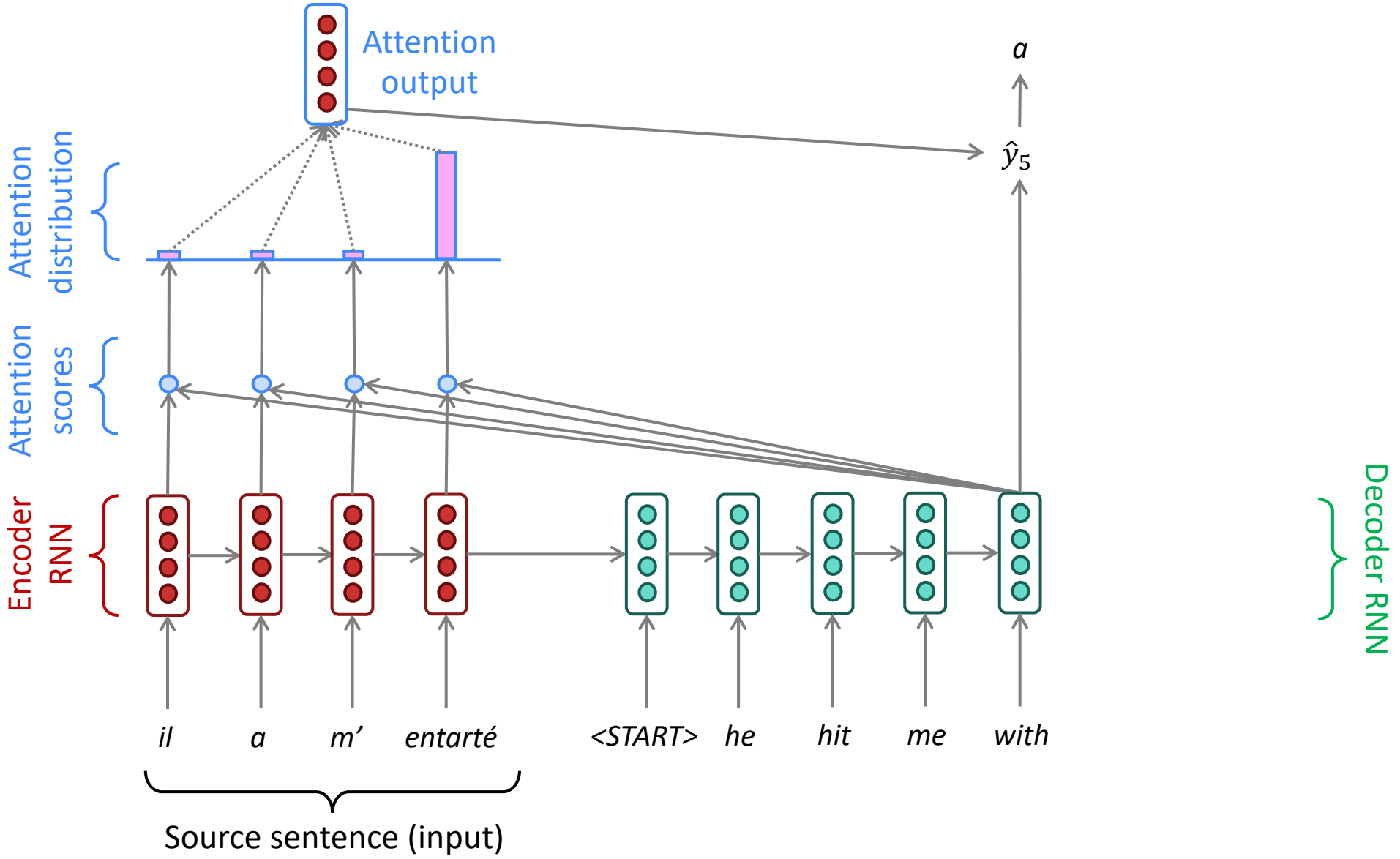
Sequence-to-sequence with attention



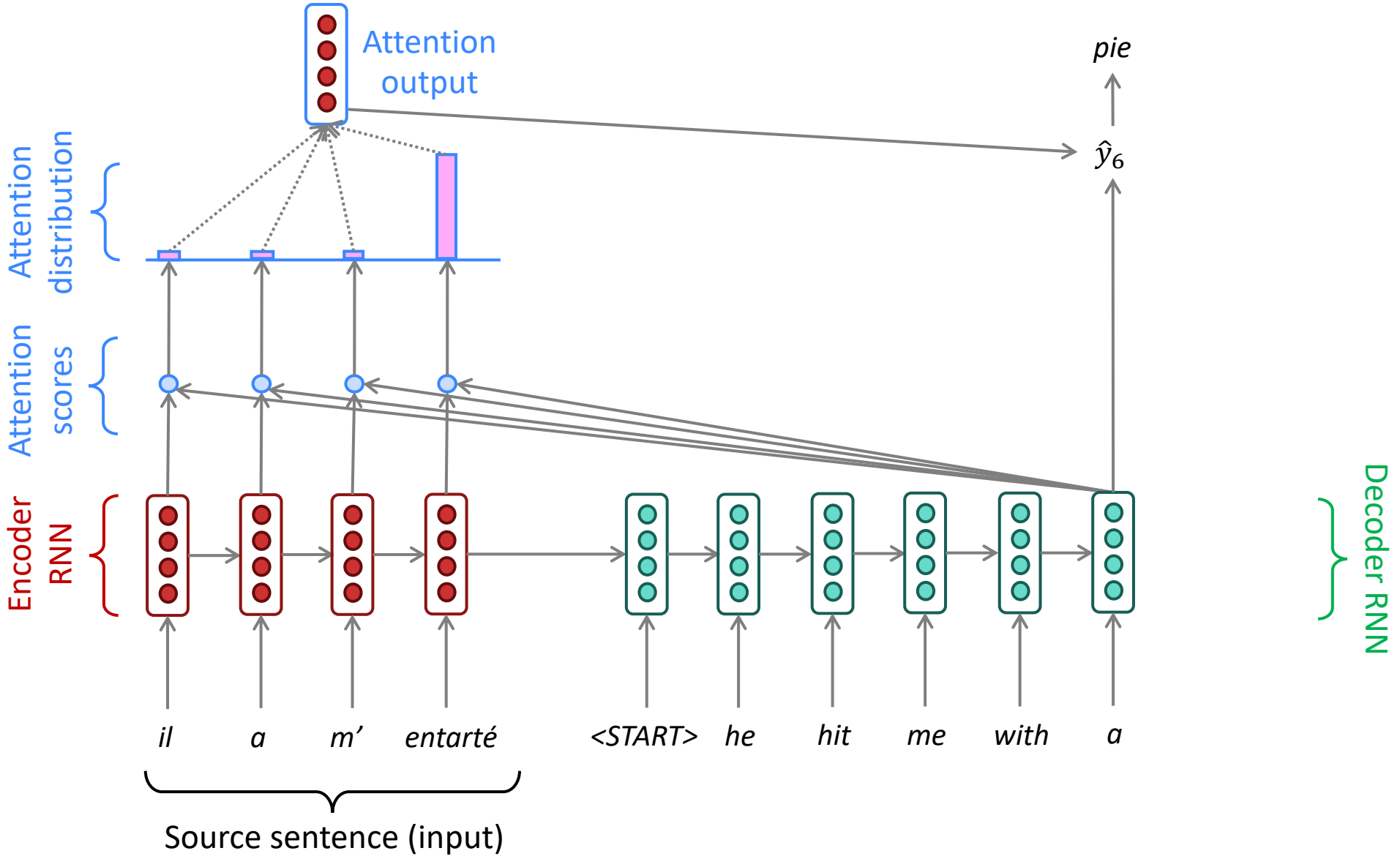
Sequence-to-sequence with attention



Sequence-to-sequence with attention



Sequence-to-sequence with attention



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)

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

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

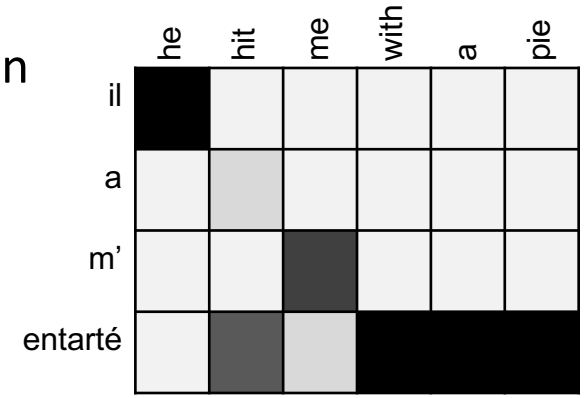
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- 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 a 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:

There are multiple ways to do this

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

  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* $\boldsymbol{a}$ (sometimes called the *context vector*)

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!

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%; poster or 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 minBERT and Downstream Tasks
    - 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 nearly all of you to keep using 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	6 PM - 8:50 PM	STLC-105, Sapp Center for Science Teaching and Learning	**Ansh Khurana** *Robustness to Distribution Shifts, CV, Healtcare*	**Heidi Zhang** *Large language models, Vision & Language*	**Eric Frankel** *Representation Learning, Dialogue Systems, Multitask*	**David Huang** *Healthcare, NLP*		
Tue	9 AM - 11:50 AM	Thronton 210, Thornton Center	**Isabel Papadimitriou** *Interpretability, Multilinguality, Language Models and Cognition*	**Shai Limonchik** *NLP, CV, ML*	**Hans Hanley** *Misinformation, Online Networks*	**Rishi Desai** *ML, CV, Privacy*	**Yuan Gao** *Diffusion/Score Based Model for Texts, CV*	**Cathy Yang** *NLP, CV*
Wed	1:30 PM - 4:20 PM	120-414, McClatchy Hall, Main Quad	**Siyan Li** *Dialogue systems, Narrative Generation, Language Generation, Speech*	**Sauren Khosla** *Healthcare, Finance*	**Drew Kaul** *ML, CV*	**Manasi Sharma** *CV, ML*	**Jesse Mu** *Language + RL, Language + Vision*	
Thu	6 PM - 8:50 PM	260-012, Pigott Hall, Main Quad	**Lisa Li** *NLP, Text Generation, Syntax, Parsing, Adaptation*	**Tathagat Verma** *Natural Language Generation, Graph Neural Networks, ML for Biomedical Imaging*	**Irena Gao** *Robustness, ML*	**Swastika Dutta** *Misinformation, Language Modelling, Security*	**John Hewitt** *Interpretability, Representation Learning, Language Modeling, Generation, Multilinguality*	
Fri	1:30 PM - 4:20 PM	110-101, Bldg.110, Main Quad	**Elaine Sui** *ML for Healthcare*	**Anuj Nagpal** *Natural Language Generation, Knowledge Graphs*	**Hong Liu** *ML*	**Christopher Cross** *ML for Climate Change, Multimodal Learning, Interpretability*		
Sat	9 AM - 11:50 AM	THORNT210, Thronton Center	**Abhinav Garg** *Healthcare, Speech Processing, Feedback Generation*	**Gabriel Poesia** *Program Generation, Formal Mathematics, Education*	**Candice Penelton** *Transfer Learning, Speech Processing, Sentiment Analysis*			

The Default Final Project

- The 2023 final project handouts will be on the website on Thursday (Feb 2)!
- New one this year: Building and experimenting with a minBERT implementation
  - Provided starter code in PyTorch. ☺
- We will discuss transformer models and BERT next in the class (next 2 lectures)
- What you do:
  - Finish writing an implementation of BERT
  - Fine-tune it for Sentiment analysis
    - Rotten Tomatoes: Light, silly, photographed with colour and depth, and rather a good time.
      Sentiment: 4 (Positive)
  - Extend and improve it in various ways of your choice:
    - Contrastive learning
    - Paraphrasing
    - Regularized optimization

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.)

Why Choose The Custom Final Project?

- If you:
  - Have some research project that you're excited about (and are possibly already working on)
  - You want to try to do something different on your own
  - 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!

- **But note: The final project for cs224n must substantively involves both human language and neural networks** (the topics of this class)**!**

Gamesmanship

- The default final project is more guided, but it should be the same amount of work
- It's just that you can focus on a given problem rather than coming up with your own
- The default final project is also an open-ended project where you can explore different approaches, but to a given problem. Strong default final projects do new things.
- There are great default final projects and great custom final projects … and there are weak default final projects and weak custom final projects.
  - The path to success is not to do something that looks kinda weak/ill-considered compared to what you could have done with the DFP.
- Neither option is the easy way to a good grade; we give Best Project Awards to both

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 14, 4:30pm on Gradescope

Project Proposal – from every team 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%

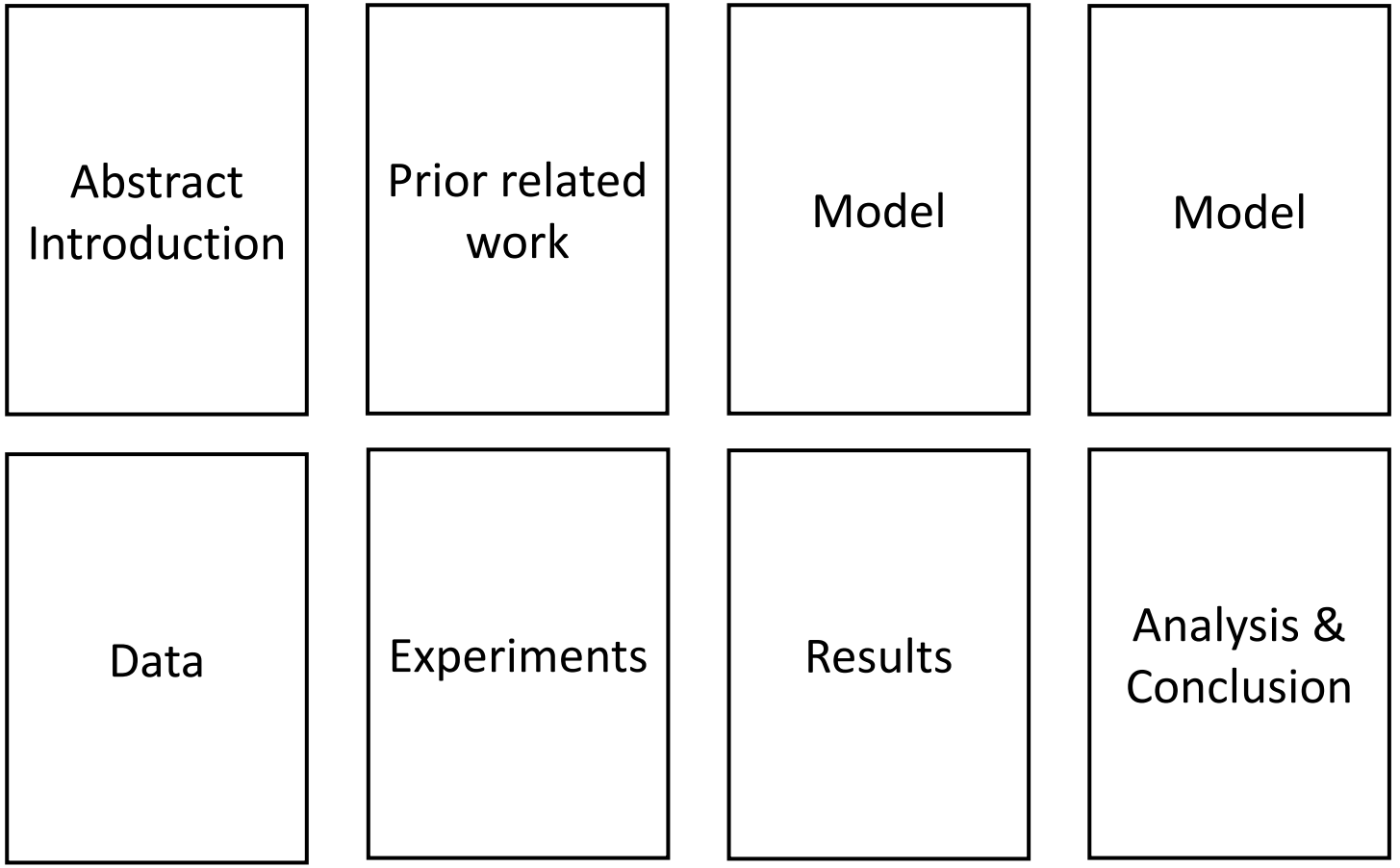
- This is a progress report
- You should be more than halfway done!
- Describe the experiments you have run
- Describe the preliminary results you have obtained
- Describe how you plan to spend the rest of your time

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 Mar 2, 4:30pm on Gradescope

Project writeup

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



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
   - All the above are expected to have experiments with numbers and ablations ☺
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 or linguistic 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

Gated LSTM

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.

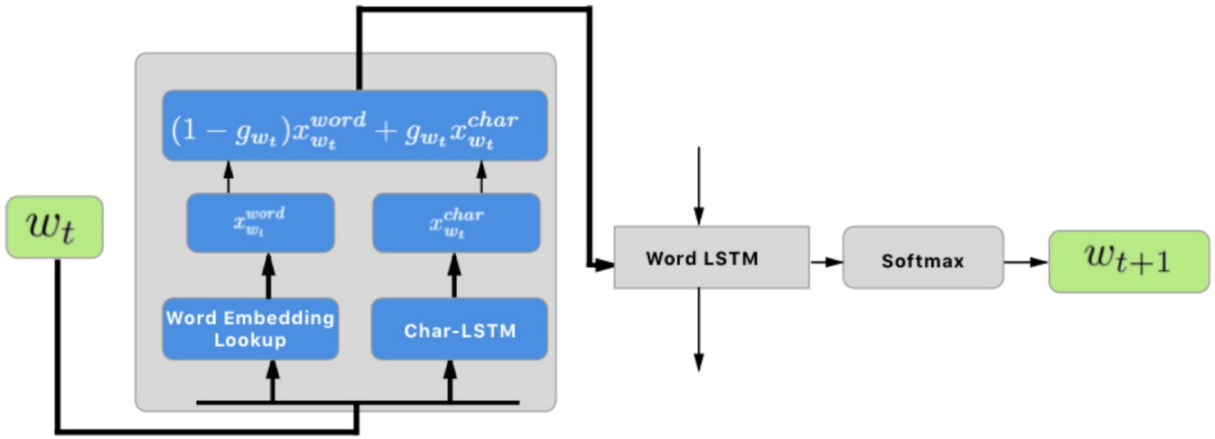
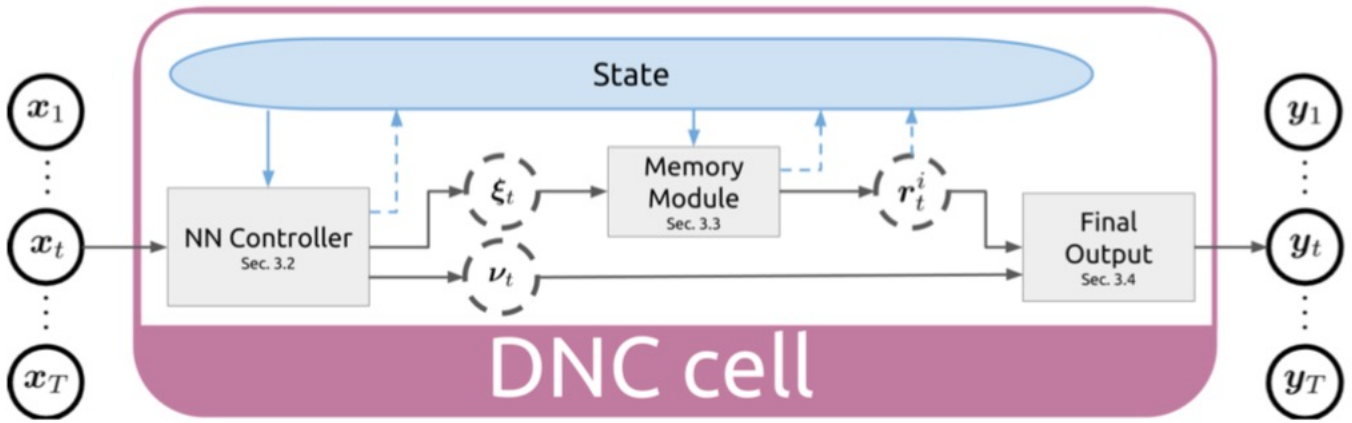


Figure 1: Architecture of the Gated LSTM

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

Enhancing Self-Consistency and Performance of Pre-Trained Language Models through Natural Language Inference

Eric Mitchell, Joseph J. Noh, Siyan Li, William S. Armstrong,
Ananth Agarwal, Patrick Liu, Chelsea Finn, Christopher D. Manning
Stanford University
eric.mitchell@cs.stanford.edu

Abstract

While large pre-trained language models are powerful, their predictions often lack logical consistency across test inputs. For example, a state-of-the-art Macaw question-answering (QA) model answers *Yes* to *Is a sparrow a bird?* and *Does a bird have feet?* but answers *No* to *Does a sparrow have feet?*. To address this failure mode, we propose a framework, Consistency Correction through Relation Detection, or **ConCoRD**, for boosting the consistency and accuracy of pre-trained NLP models using pre-trained natural language inference (NLI) models without fine-tuning or re-training. Given a batch of test inputs, ConCoRD samples several candidate outputs for each input and instantiates a factor graph that accounts for both the model's belief about the likelihood of each answer choice in isolation and the NLI model's beliefs about pair-wise answer choice compatibility. We show that a weighted MaxSAT solver can efficiently compute high-

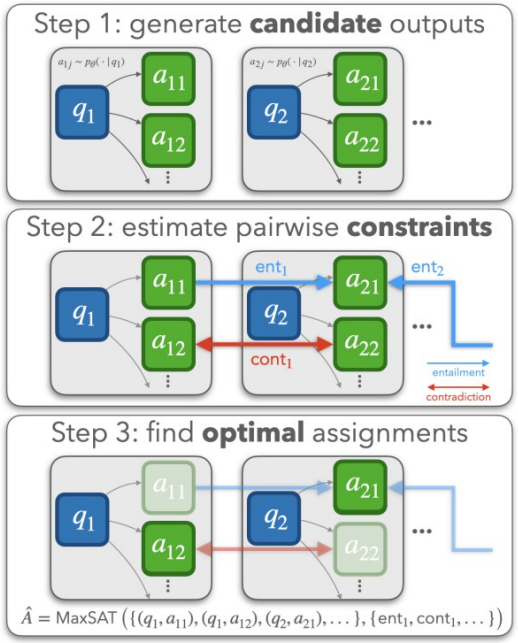


Figure 1: ConCoRD first generates candidate outputs from the base pre-trained model, then estimates soft pairwise constraints between output choices, and finally finds the most satisfactory choices of answers accounting for both the base model and NLI model's beliefs.

Merge of the work of two CS224N project teams, published at EMNLP 2022

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.

4. 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
  - http://cs224n.stanford.edu/
- 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 2019–2023, 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.

2023 NLP … recommended for all your practical projects ☺

pip install transformers # By **Huggingface** 🤗 Tutorial: Fri Feb 10, 3:30–4:20, Gates B01

```
# 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 2023g

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
  - Adaptation when there is 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 2023

- 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
    - COGS: https://github.com/najoungkim/COGS
    - gSCAN: https://arxiv.org/abs/2003.05161

Can I use train GPT-2 or use ChatGPT to do my final project?

- You need to be very cognizant of how large a model you can train
  - You just don't have the resources to train your own GPT-2 model
  - You probably don't have the resources to even **load** T5 11B

- You are welcome to use GPT-3 or ChatGPT in your final project
  - Though we can't fund your API use bills
- There are almost certainly interesting projects that you can do using them
  - Analysis projects
  - Learning good prompts
  - Chain of thought reasoning
- But be careful to remember that you will be evaluated based on what you have done – and not on the amazing ChatGPT output that you show us ("look, it works zero shot")

5. Finding data for your projects

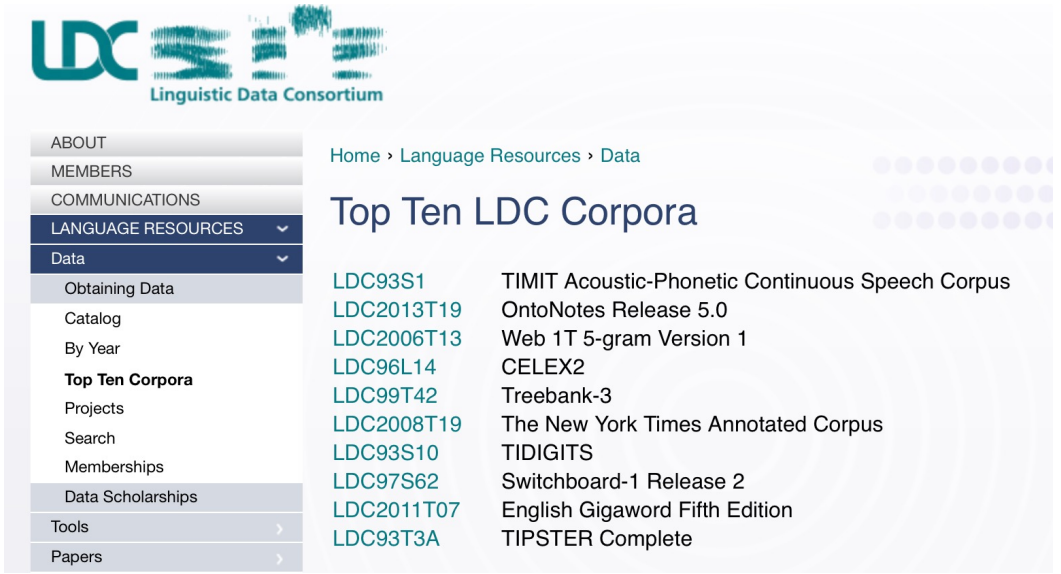
- Some people collect their own data for a project – **we like that, but it's trick to do fast!**
  - You may have a project that uses "unsupervised" data
  - You can annotate a small amount of data
  - You can find a website that effectively provides annotations, such as likes, stars, ratings, responses, etc.
    - This 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

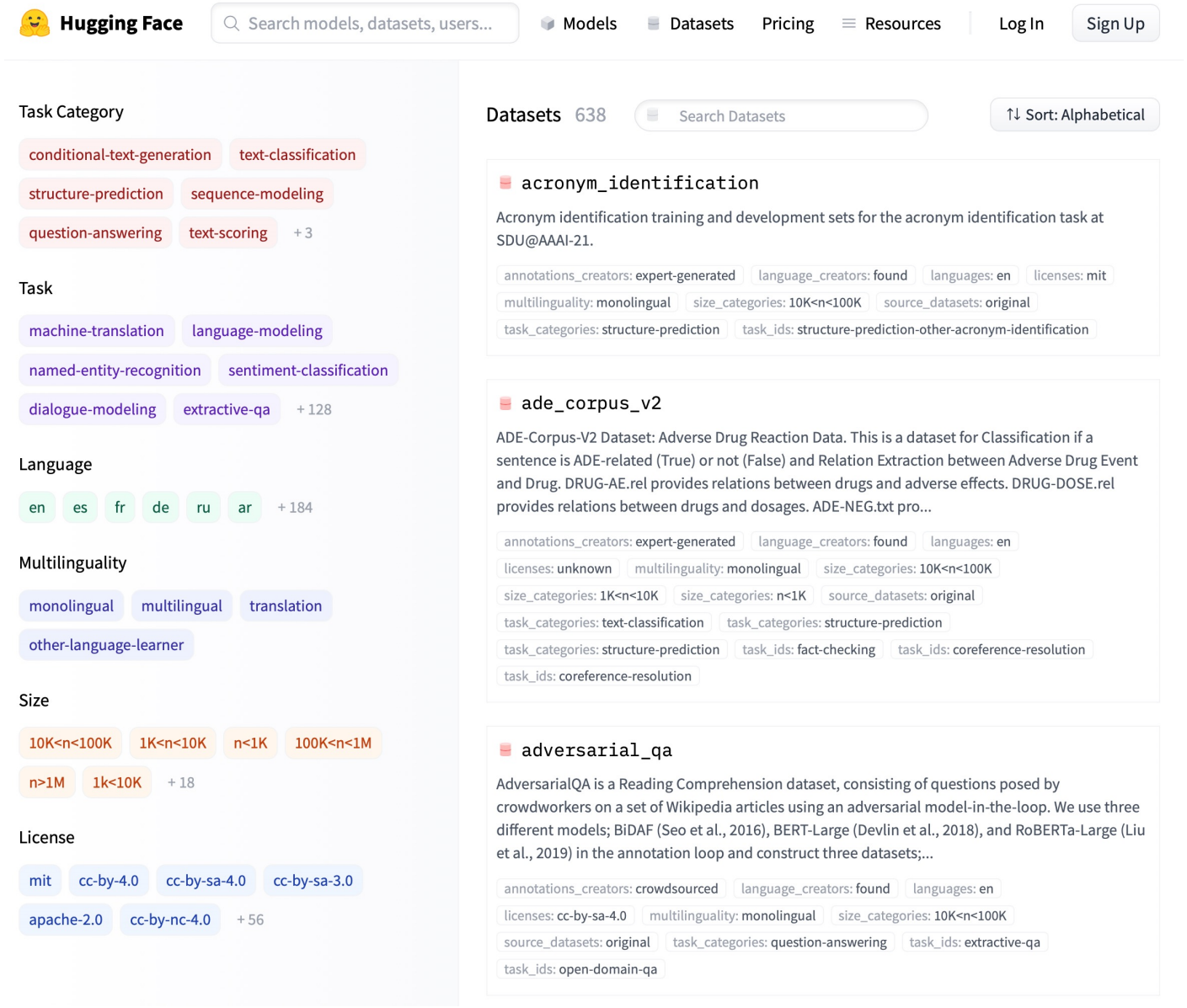
Linguistic Data Consortium

- https://catalog.ldc.upenn.edu/
- Stanford licenses this data; you can get access. Sign up/ask questions 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!



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 (General Language Understanding Evaluation benchmark)

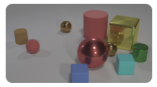
General Language Understanding Evaluation (GLUE) benchmark is a collection of nine natural language understanding tasks, including single-sentence tasks CoLA and SST-2, similarity...

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

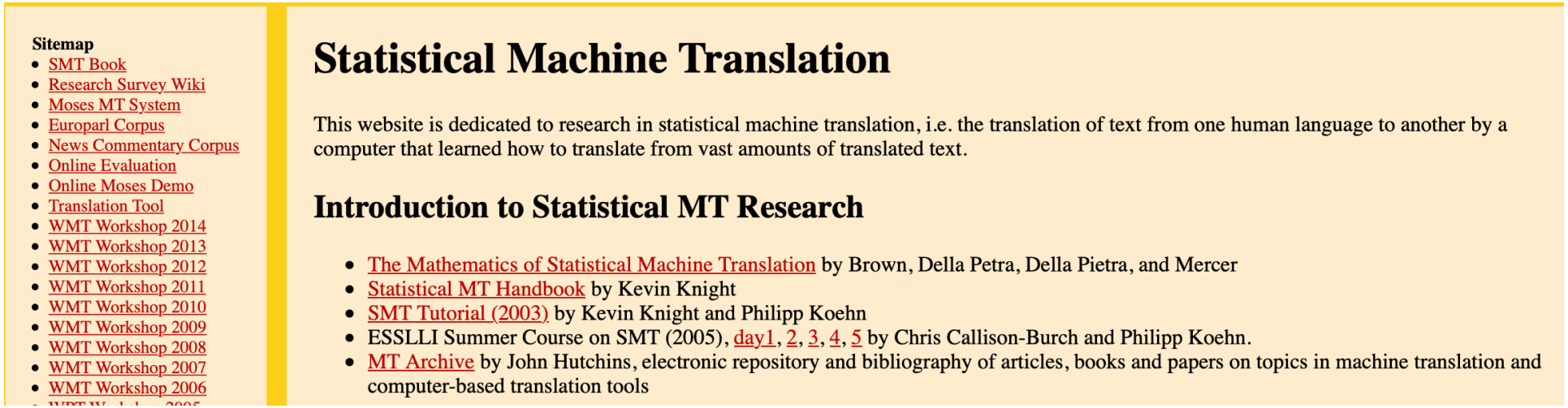
Billion Word Benchmark

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

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.

Many, many more

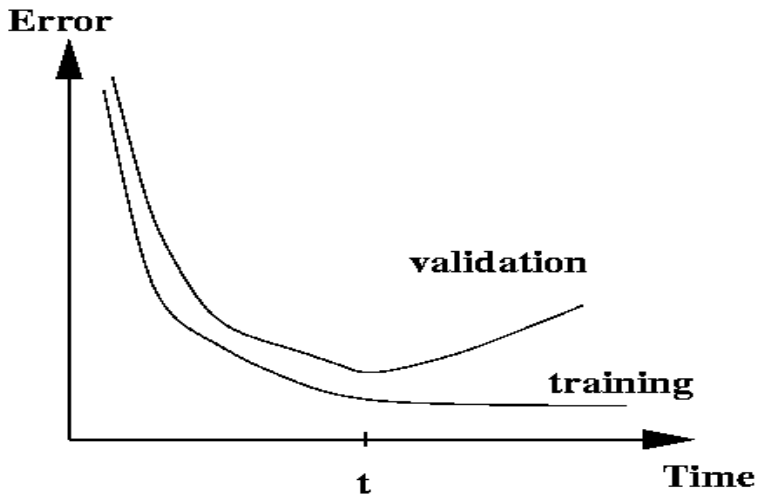
- There are now many other datasets available online for all sorts of purposes
  - Look at Kaggle
  - Look at research papers to see what data they use
  - Look at lists of datasets
    - https://machinelearningmastery.com/datasets-natural-language-processing/
    - https://github.com/niderhoff/nlp-datasets
  - 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

5. 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 neural nets

Good luck with your projects!