Natural Language Processing with Deep Learning

CS224N/Ling284

Christopher Manning

Lecture 9: Practical Tips for Final Projects

Lecture Plan

Lecture 9: Practical Tips for Final Projects – A pause for breath!

1. Final project types and details; assessment revisited
2. Finding research topics; a couple of examples
3. Finding data
4. Review of gated neural sequence models
5. A couple of MT topics
6. Doing your research
7. Presenting your results and evaluation

1. 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: 3%; report: 30%
  - Final poster session attendance expected! (See website.)
    Wed Mar 20, 5pm-10pm (put it in your calendar!)
- Participation: 3%
  - Guest/random lecture attendance, Piazza, eval, karma – see website!
- Late day policy
  - 6 free late days; then 10% 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

Mid-quarter feedback survey

- Is going out today
- Please fill it in!
- We'd love to get your thoughts on the course so far!
- A good chance to improve the course immediately, as well as helping for future years
- Bribe: 0.5% participation points – make sure to submit the second form that records your name disassociated from the survey

The Final Project

- For FP, you either
  - Do the default project, which is SQuAD question answering
    - 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
  - Larger team project or a project for multiple classes should be larger and often involve exploring more tasks

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

The Default Final Project

- Materials will be released on Thursday
- Task: Building a textual question answering system for SQuAD
  - Stanford Question Answering Dataset
    - https://rajpurkar.github.io/SQuAD-explorer/
  - New this year:
    - Providing starter code in PyTorch ☺
    - Attempting SQuAD 2.0 rather than SQuAD 1.1 (has unanswerable Qs)

- I will discuss question answering and SQuAD in Thursday's class

T: [Bill] Aken, 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!

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

This lecture is still relevant ... Even if doing DFP

- At a lofty level
  - It's good to know something about how to do research!

- At a prosaic level
  - We'll touch on:
    - Baselines
    - Benchmarks
    - Evaluation
    - Error analysis
    - Paper writing

    which are all great things to know about for the DFP too!

Why Choose The Custom Final Project?

- If you:
  - Have some research project that you're excited about (and possibly already working on)
  - 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)
  - 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!

Project Proposal – from everyone 5%

1. Find a relevant research paper for your topic
   - For DFP, a paper on the SQuAD leaderboard will do, but you might look elsewhere for interesting QA/reading comprehension work
2. Write a summary of that research paper and describe how you hope to use or adapt ideas from it and how you plan to extend or improve it in your final project work
   - Suggest a good milestone to have achieved as a halfway point
3. 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

2–4 pages. Details released this Thursday

Due Thu Feb 14, 4:30pm on Gradescope

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 7, 4:30pm on Gradescope

2. 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 approach of interest, and work out good ways to extend or improve it or 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, usually applying an existing neural network model
2. Implement a complex neural architecture and demonstrate its performance on some data
3. Come up with a new or variant neural network model 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

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.

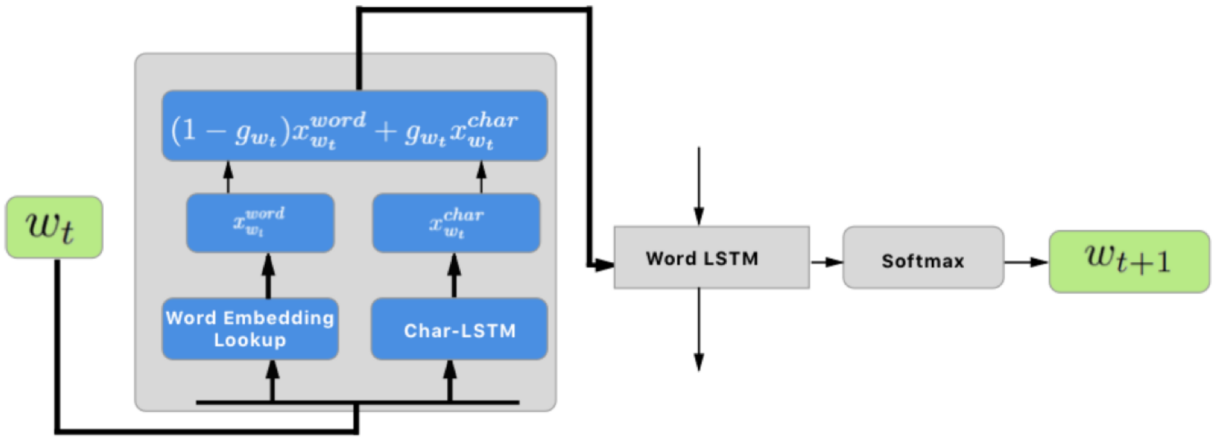
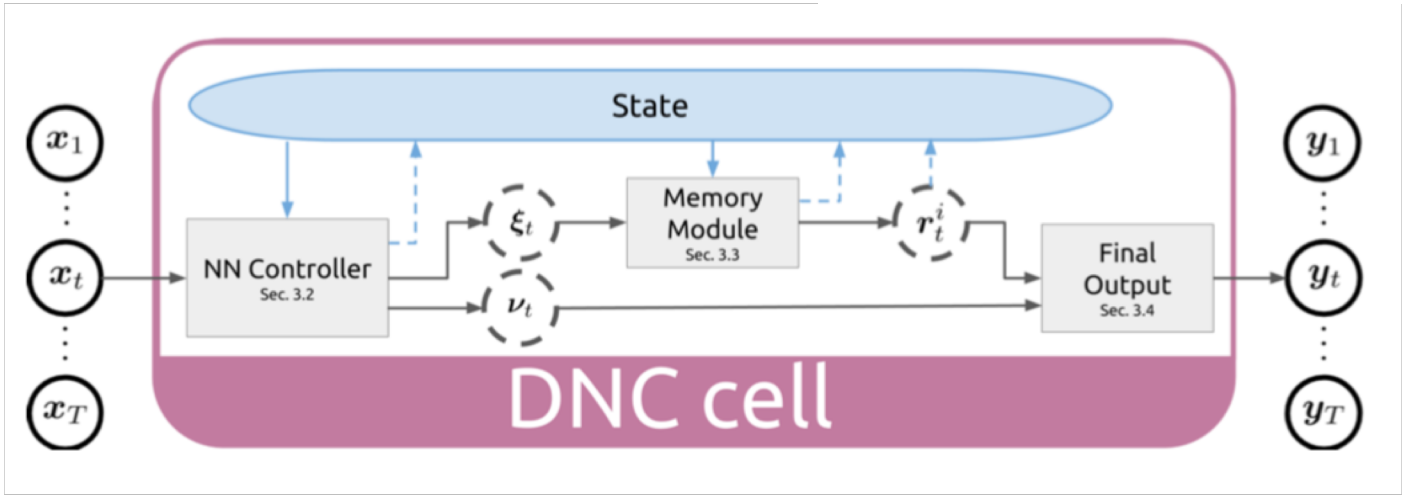


Figure 1: Architecture of the Gated LSTM

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

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.info
- Also look at the online proceedings of major ML conferences:
  - NeurIPS, ICML, ICLR
- Look at past cs224n project
  - See the class website
- Look at online preprint servers, especially:
  - https://arxiv.org

- Even better: look for an interesting problem in the world

How to find an interesting place to start?

Arxiv Sanity Preserver by Stanford grad Andrej Karpathy of cs231n

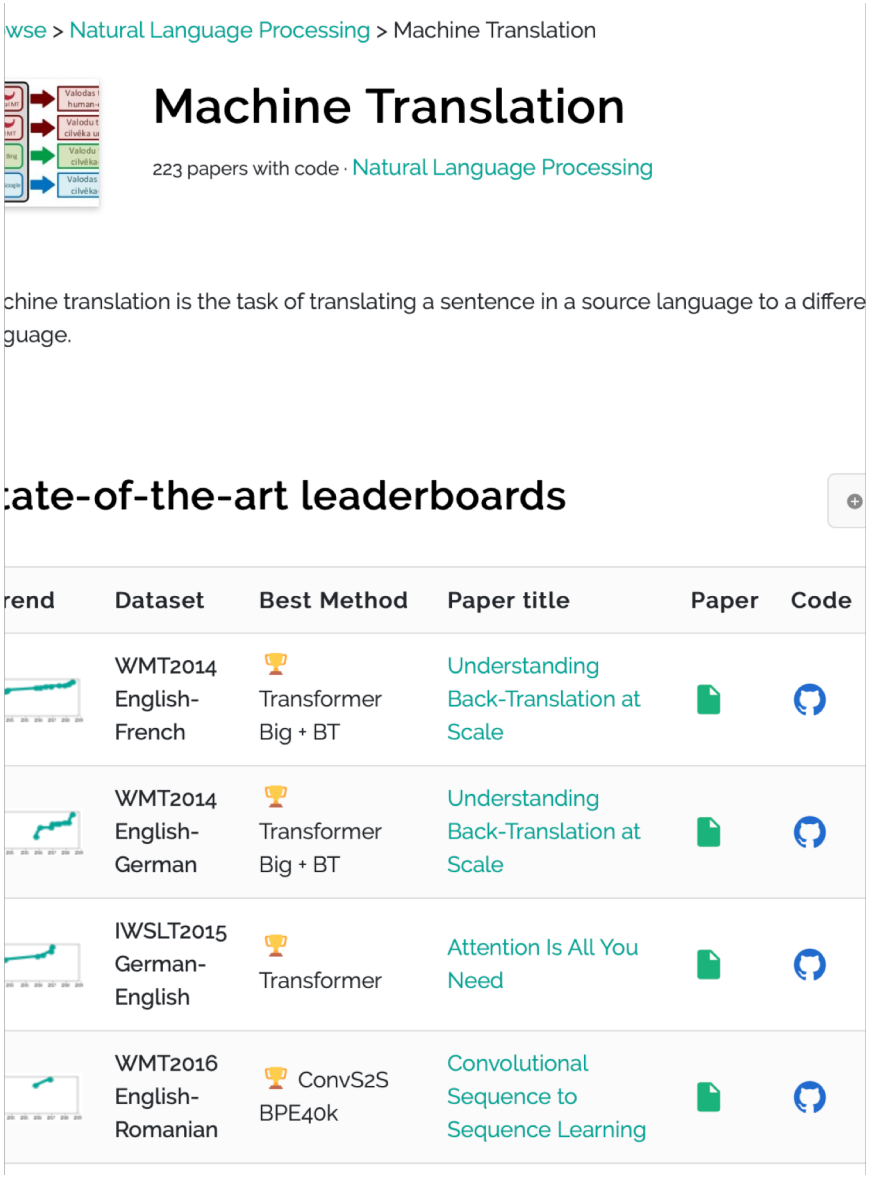
http://www.arxiv-sanity.com

Want to beat the state of the art on something?

Great new site – a much needed resource for this – lots of NLP tasks

- Not always correct, though

https://paperswithcode.com/sota



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.”

Must-haves (for most* custom final projects)

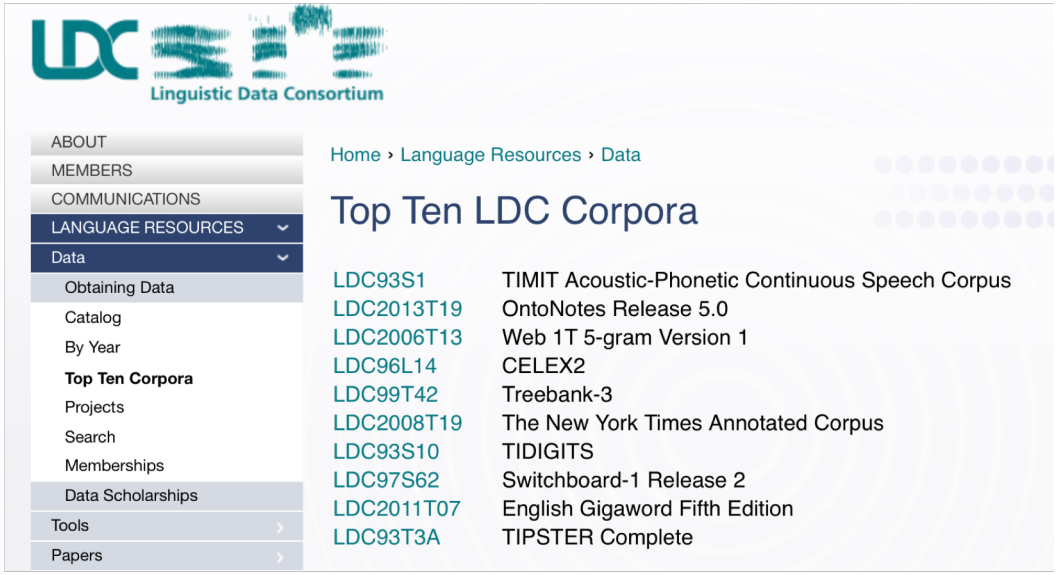
- Suitable data
  - Usually aiming at: 10,000+ labeled examples by milestone
- Feasible task
- Automatic evaluation metric
- NLP is central to the project

3. Finding data

- Some people collect their own data for a project
  - 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, etc.
    - Let’s you learn about real word challenges of applying ML/NLP!
- 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 an existing, curated dataset built by previous researchers**
    - You get a fast start and there is obvious prior work and baselines

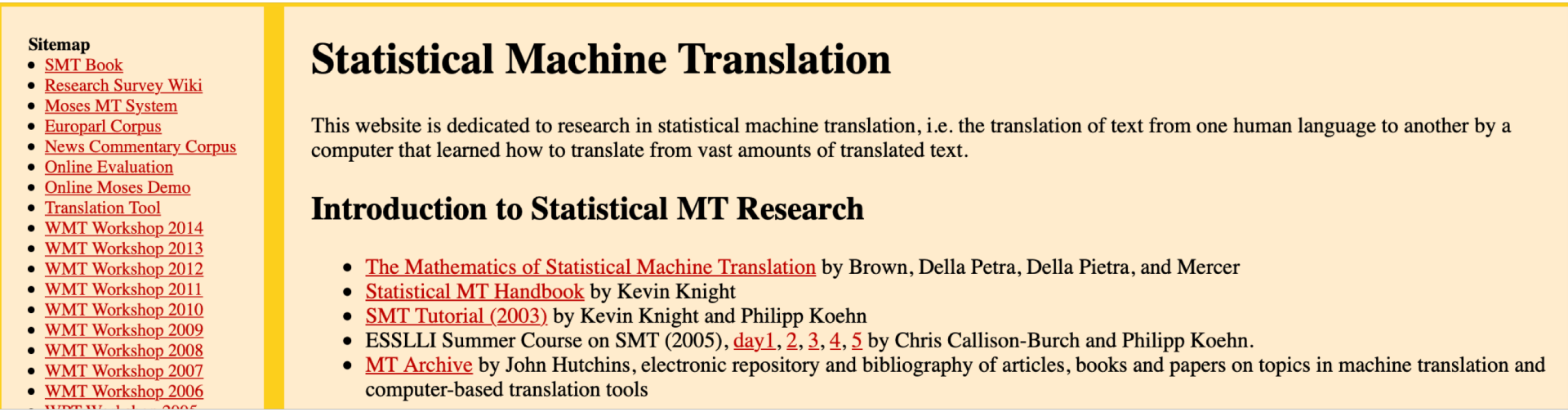
Linguistic Data Consortium

- 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 newswire, lots of speech with transcription, parallel MT data
  - 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.

Many, many more

- There are now many other datasets available online for all sorts of purposes
  - Look at Kaggle
  - Look at research papers
  - Look at lists of datasets
    - https://machinelearningmastery.com/datasets-natural-language-processing/
    - https://github.com/niderhoff/nlp-datasets
  - Ask on Piazza or talk to course staff

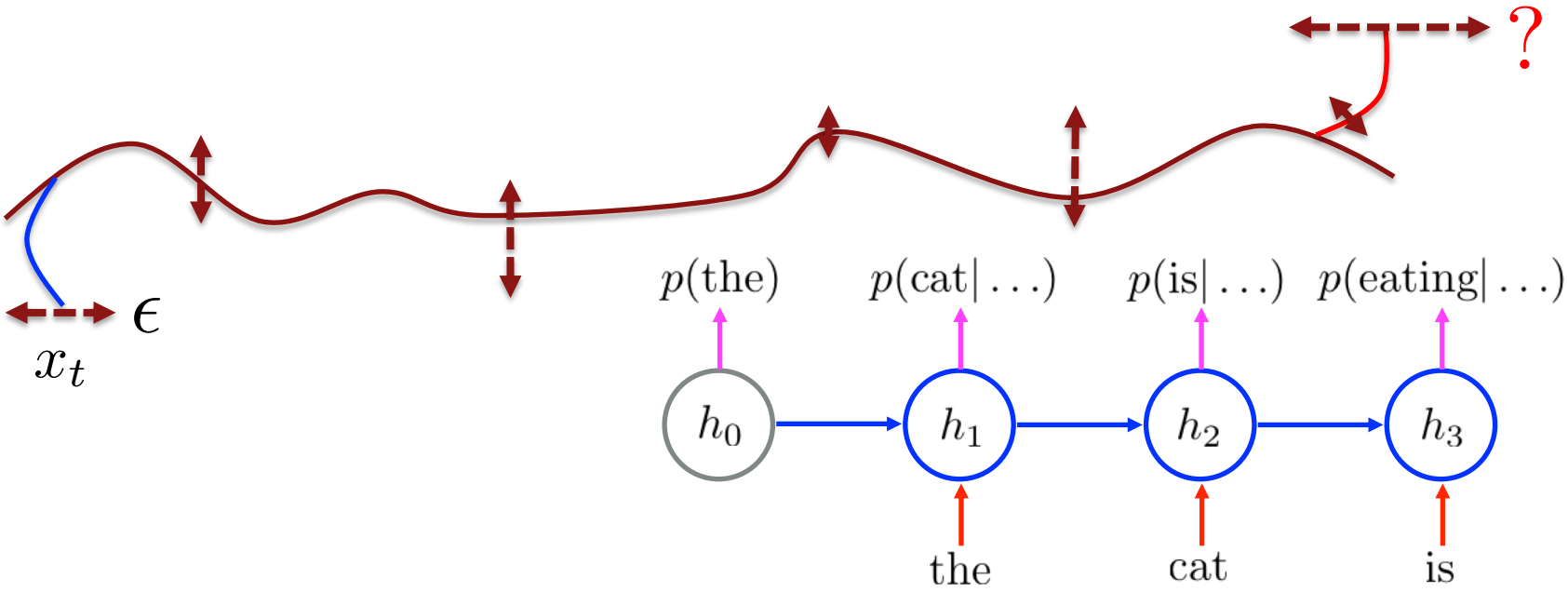
4. One more look at gated recurrent units and MT

Intuitively, what happens with RNNs?

1. Measure the influence of the past on the future

$$\frac{\partial \log p(x_{t+n}|x_{<t+n})}{\partial h_t} = \frac{\partial \log p(x_{t+n}|x_{<t+n})}{\partial g} \frac{\partial g}{\partial h_{t+n}} \frac{\partial h_{t+n}}{\partial h_{t+n-1}} \cdots \frac{\partial h_{t+1}}{\partial h_t}$$

2. How does the perturbation at t affect $p(x_{t+n}|x_{<t+n})$?



Backpropagation through Time

Problem: Vanishing gradient is super-problematic

- When gradient goes to zero, we cannot tell whether
  1. No dependency between *t* and *t+n* in data, or
  2. Wrong configuration of parameters (the vanishing gradient condition)
- Is the problem with the naïve transition function?

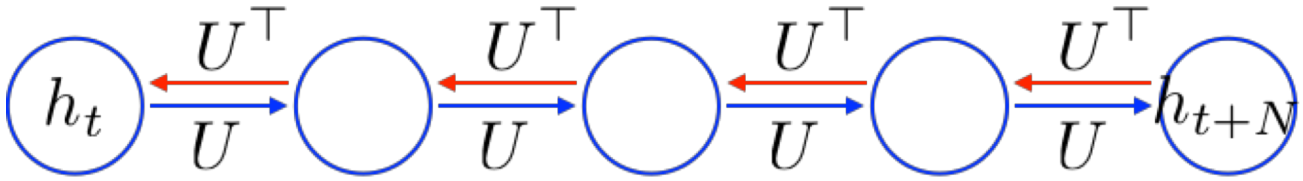
$$f(h_{t-1}, x_t) = \tanh(W\,[x_t] + U h_{t-1} + b)$$

- With it, the temporal derivative leads to vanishing

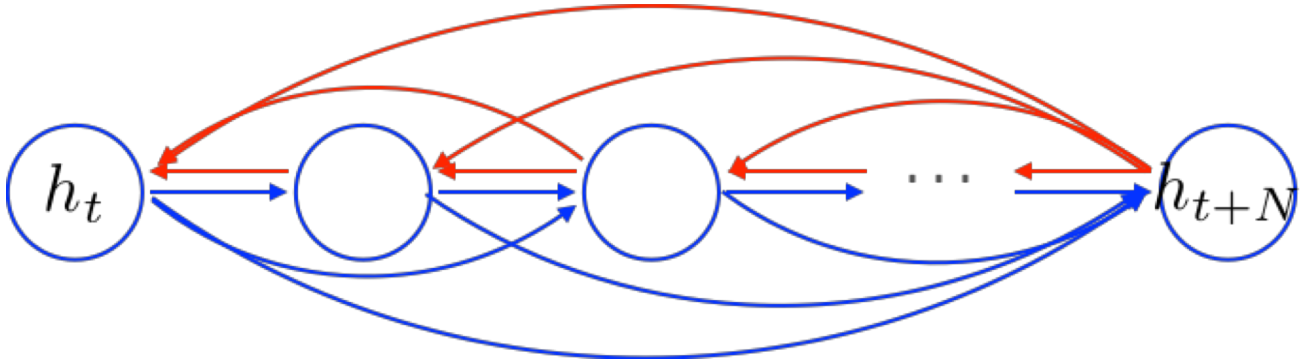
$$\frac{\partial h_{t+1}}{\partial h_t} = U^\top \frac{\partial \tanh(a)}{\partial a}$$

Gated Recurrent Unit

- It implies that the error must backpropagate through all the intermediate nodes:

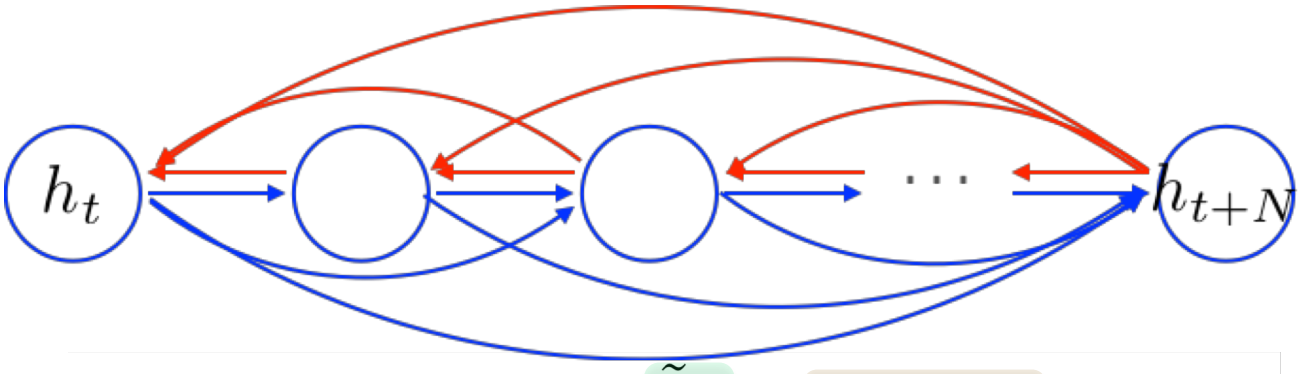


- Perhaps we can create shortcut connections.



Gated Recurrent Unit

- Perhaps we can create *adaptive* shortcut connections.



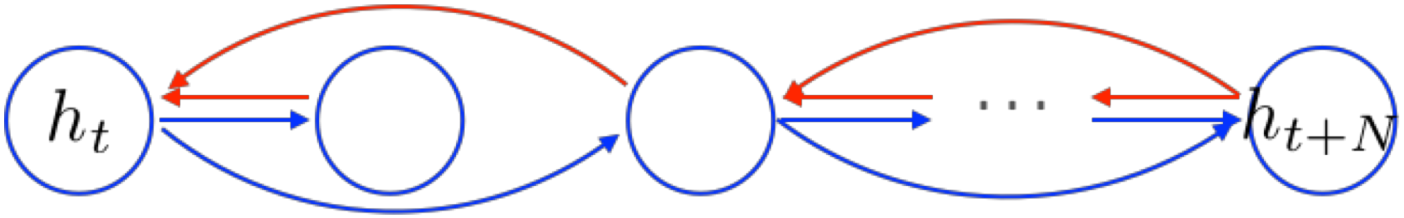
$$f(h_{t-1}, x_t) = u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1}$$

- Candidate Update $\tilde{h}_t = \tanh(W\,[x_t] + U h_{t-1} + b)$
- Update gate $u_t = \sigma(W_u\,[x_t] + U_u h_{t-1} + b_u)$

$\odot$: element-wise multiplication

Gated Recurrent Unit

- Let the net prune unnecessary connections *adaptively*.

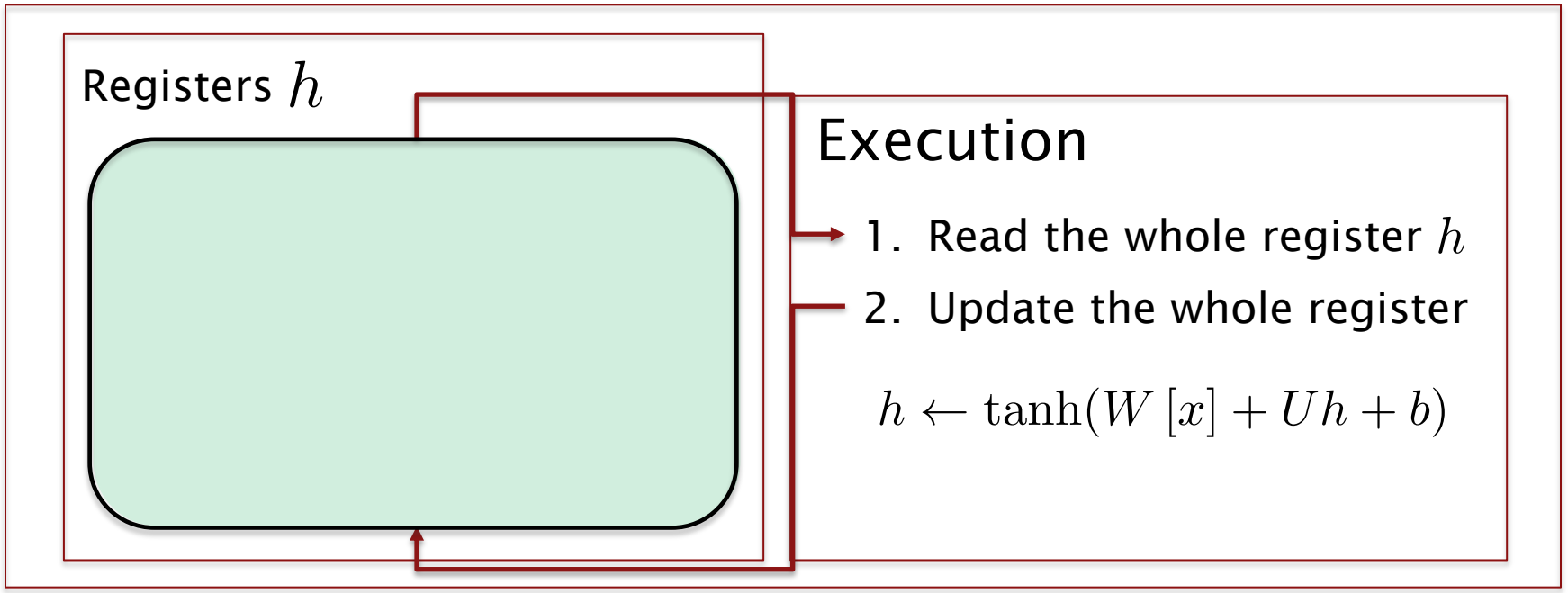


$$f(h_{t-1}, x_t) = u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1}$$

- Candidate Update $\tilde{h}_t = \tanh(W\,[x_t] + U(r_t \odot h_{t-1}) + b)$
- Reset gate $r_t = \sigma(W_r\,[x_t] + U_r h_{t-1} + b_r)$
- Update gate $u_t = \sigma(W_u\,[x_t] + U_u h_{t-1} + b_u)$

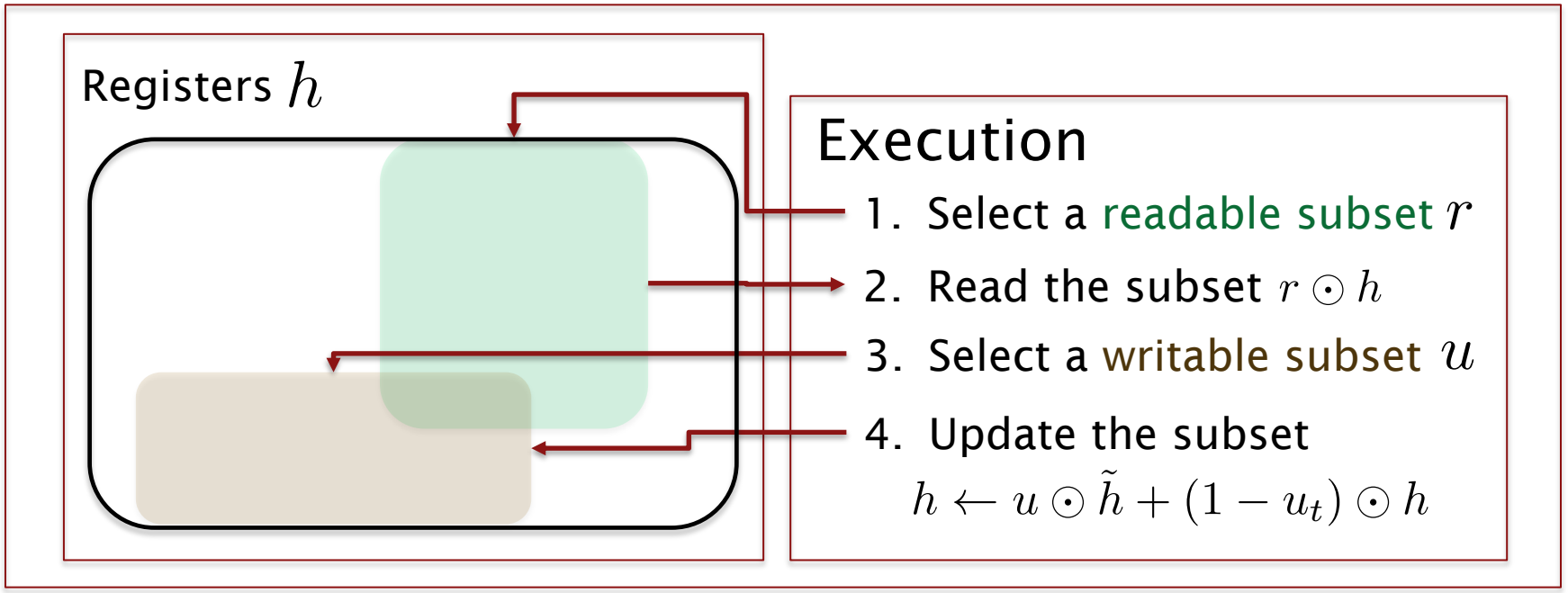
Gated Recurrent Unit

tanh-RNN ….



Gated Recurrent Unit

GRU …



Gated recurrent units are much more realistic!

Note that there is some overlap in ideas with attention

Gated Recurrent Units

Two most widely used gated recurrent units: GRU and LSTM

Gated Recurrent Unit

[Cho et al., EMNLP2014; Chung, Gulcehre, Cho, Bengio, DLUFL2014]

$$h_t = u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1}$$

$$\tilde{h}_t = \tanh(W\,[x_t] + U(r_t \odot h_{t-1}) + b)$$

$$u_t = \sigma(W_u\,[x_t] + U_u h_{t-1} + b_u)$$

$$r_t = \sigma(W_r\,[x_t] + U_r h_{t-1} + b_r)$$

Long Short-Term Memory

[Hochreiter & Schmidhuber, NC1999; Gers, Thesis2001]

$$h_t = o_t \odot \tanh(c_t)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

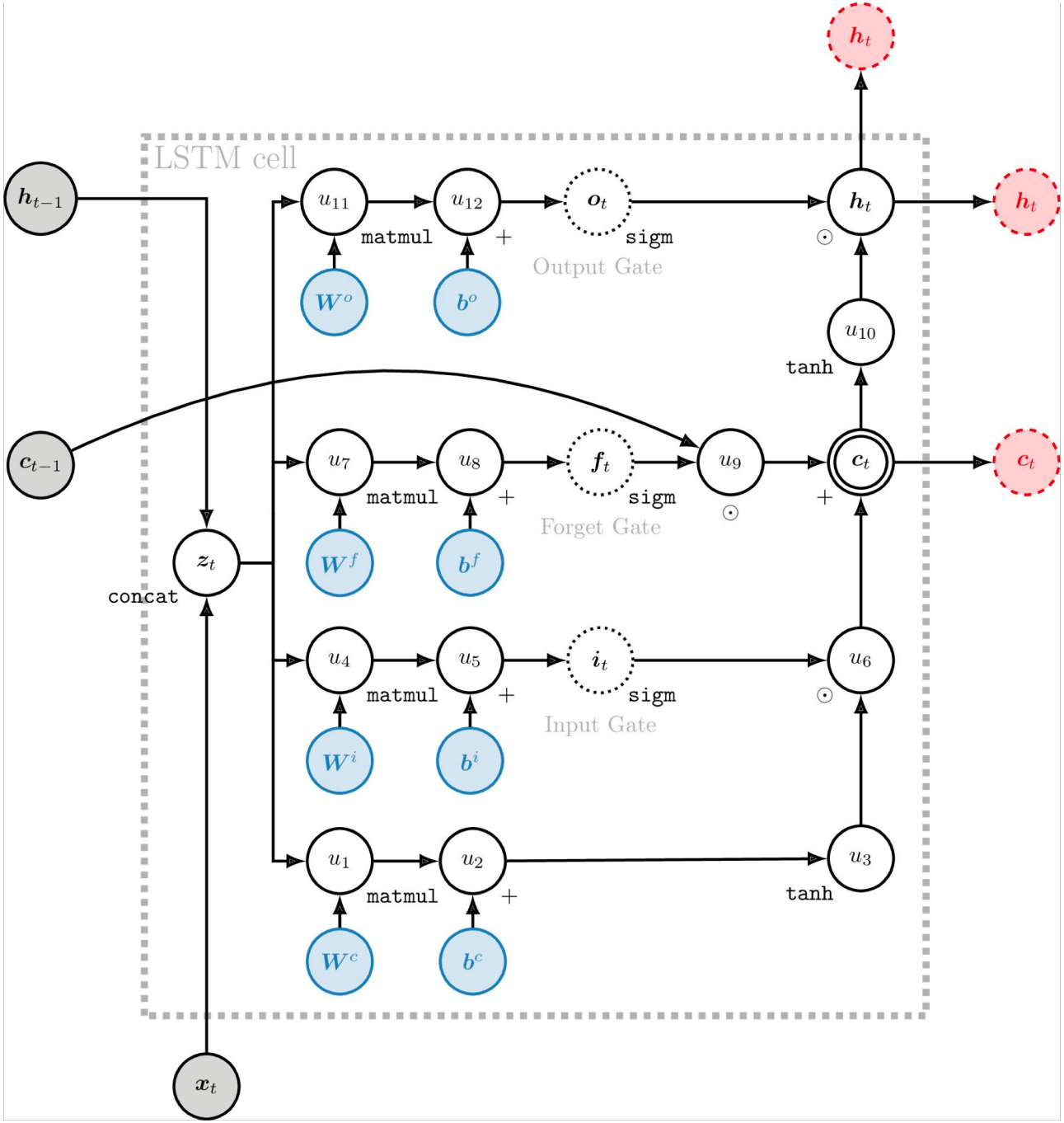
$$\tilde{c}_t = \tanh(W_c\,[x_t] + U_c h_{t-1} + b_c)$$

$$o_t = \sigma(W_o\,[x_t] + U_o h_{t-1} + b_o)$$

$$i_t = \sigma(W_i\,[x_t] + U_i h_{t-1} + b_i)$$

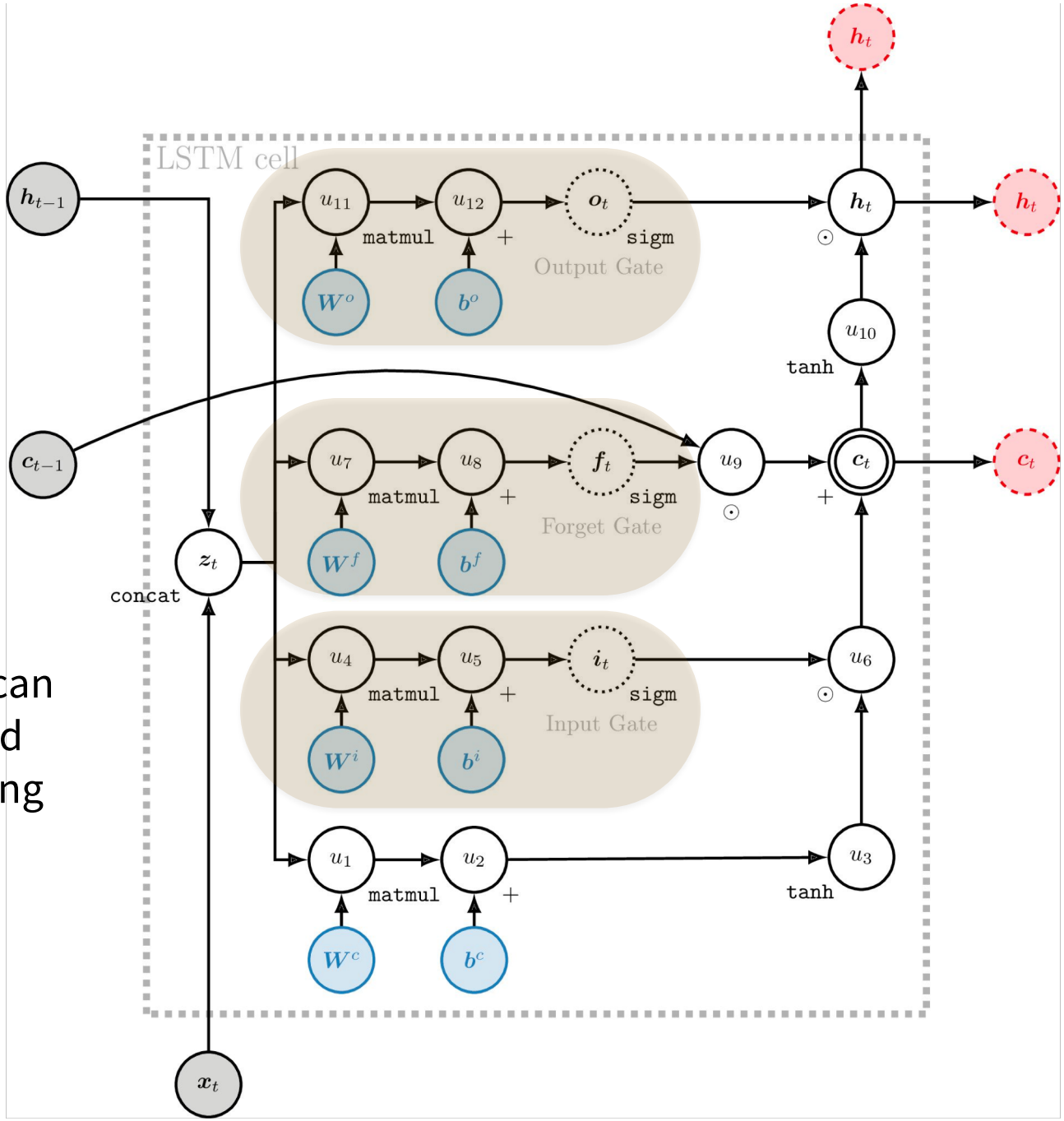
$$f_t = \sigma(W_f\,[x_t] + U_f h_{t-1} + b_f)$$

The LSTM

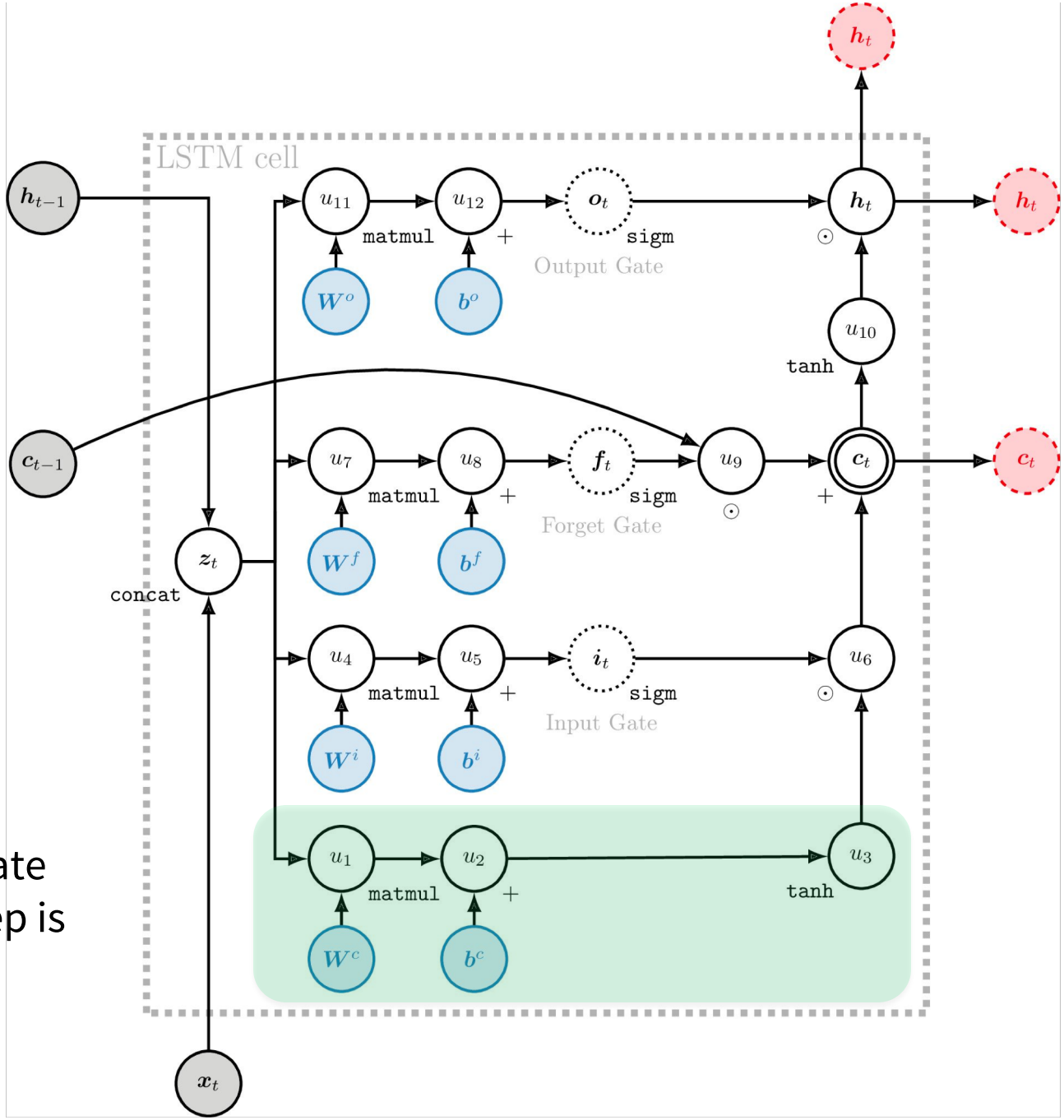

LSTM cell
h_{t-1}
c_{t-1}
x_t
z_t
concat
u_1
u_2
u_3
matmul
tanh
W^c
b^c
u_4
u_5
u_6
i_t
sigm
Input Gate
W^i
b^i
u_7
u_8
u_9
f_t
Forget Gate
W^f
b^f
u_{10}
u_{11}
u_{12}
o_t
Output Gate
W^o
b^o
c_t
h_t

The LSTM

The LSTM gates all operations so stuff can be forgotten/ignored rather than it all being crammed on top of everything else

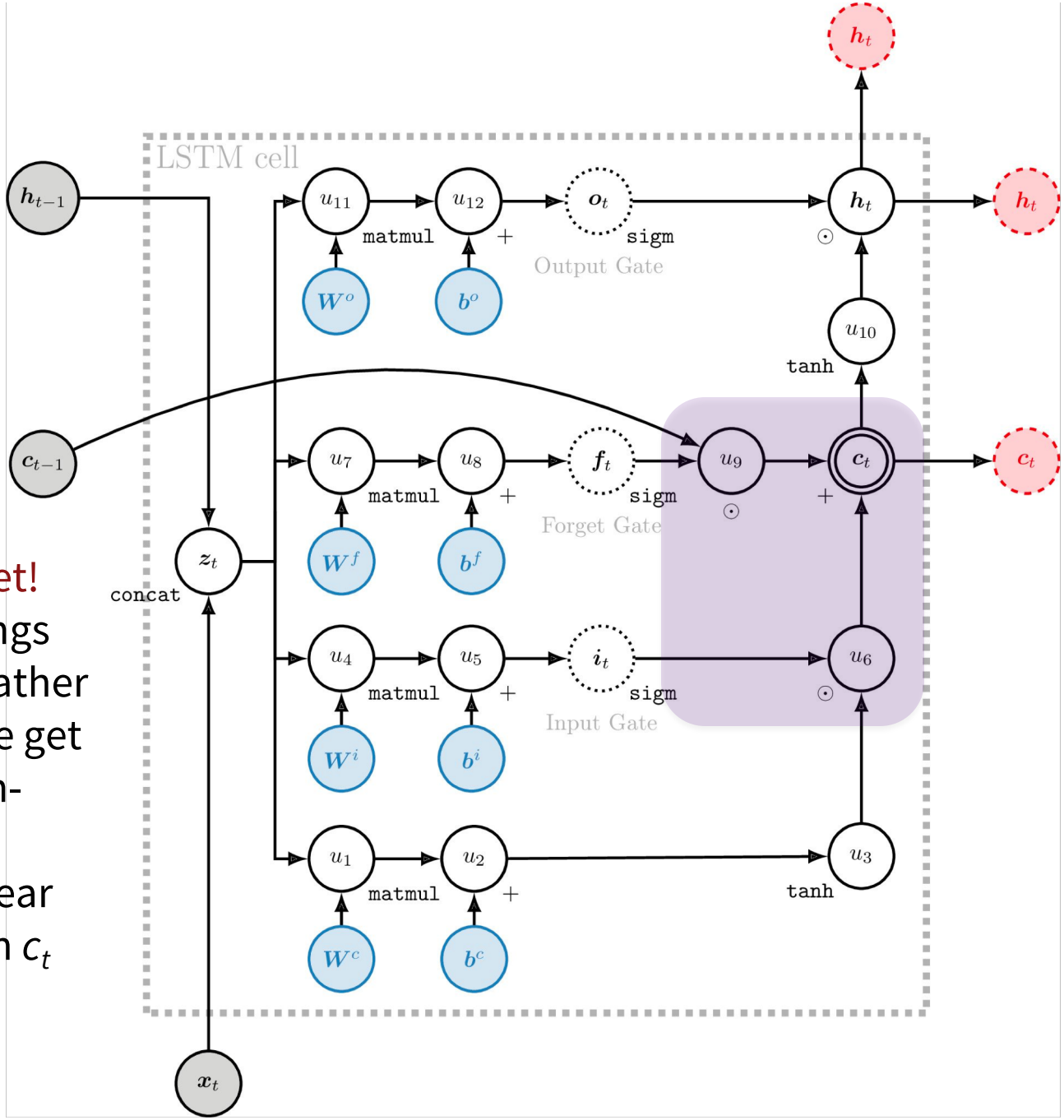


The LSTM



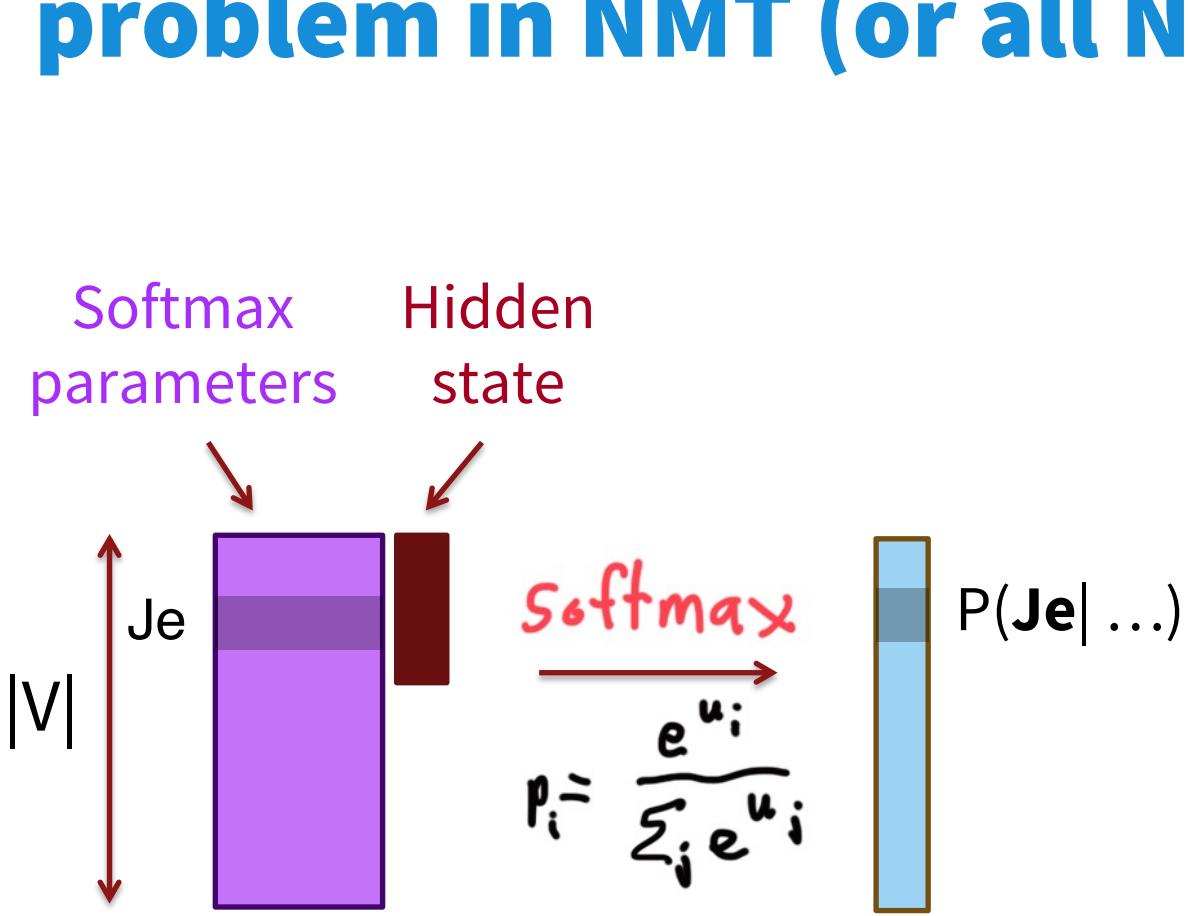
The non-linear update for the next time step is just like an RNN

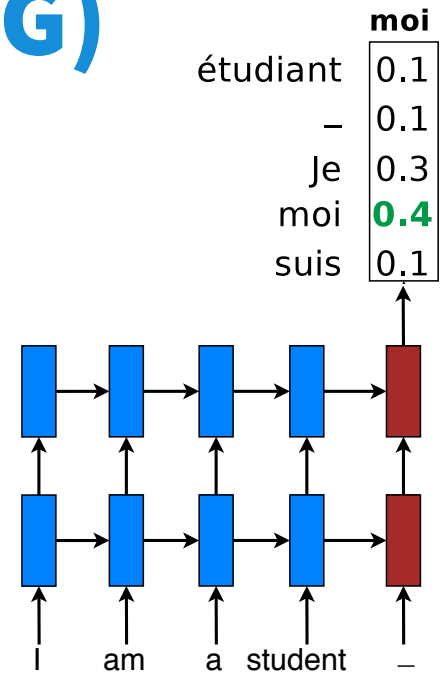
The LSTM



This part is the secret!
(Of other recent things like ResNets too!) Rather than multiplying, we get c_t by adding the non-linear stuff and c_{t-1} !
There is a direct, linear connection between c_t and c_{t-1}.

5. The large output vocabulary problem in NMT (or all NLG)

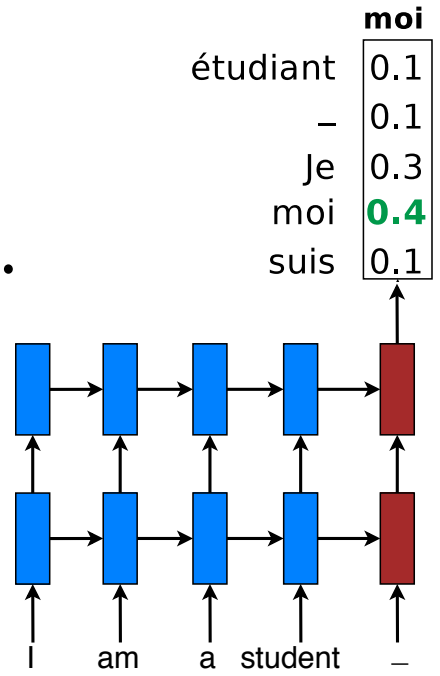




Softmax computation is expensive.

The word generation problem

- Word generation problem
  - Vocabs used are usually modest: 50K.



The ecotax portico in Pont-de-Buis
Le portique écotaxe de Pont-de-Buis

The <unk> portico in <unk>
Le <unk> <unk> de <unk>

Possible approaches for output

- *Hierarchical softmax*: tree-structured vocabulary
- *Noise-contrastive estimation*: binary classification
- *Train* on a subset of the vocabulary at a time; *test* on a smart on the set of possible translations
  - *Jean, Cho, Memisevic, Bengio. ACL2015*
- *Use attention to work out what you are translating:* You can do something simple like dictionary lookup
- *More ideas we will get to:* Word pieces; char. models

MT Evaluation – an example of eval

- Manual (the best!?):
  - **Adequacy and Fluency** (5 or 7 point scales)
  - Error categorization
  - **Comparative ranking of translations**

- Testing in an application that uses MT as one sub-component
  - E.g., question answering from foreign language documents
    - May not test many aspects of the translation (e.g., cross-lingual IR)

- Automatic metric:
  - **BLEU (Bilingual Evaluation Understudy)**
  - Others like TER, METEOR, …

BLEU Evaluation Metric

(Papineni et al, ACL-2002)

Reference (human) translation:

The U.S. island of Guam is maintaining a high state of alert after the Guam airport and its offices both received an e-mail from someone calling himself the Saudi Arabian Osama bin Laden and threatening a biological/chemical attack against public places such as the airport .

Machine translation:

The American [?] international airport and its the office all receives one calls self the sand Arab rich business [?] and so on electronic mail, which sends out ; The threat will be able after public place and so on the airport to start the biochemistry attack , [?] highly alerts after the maintenance.

- N-gram precision (score is between 0 & 1)
  - What percentage of machine n-grams can be found in the reference translation?
    - An n-gram is an sequence of n words
  - Not allowed to match same portion of reference translation twice at a certain n-gram level (two MT words *airport* are only correct if two reference words *airport;* can't cheat by typing out "the the the the the")
  - Do count unigrams also in a bigram for unigram precision, etc.
- Brevity Penalty
  - Can't just type out single word "the" (precision 1.0!)
- It was thought quite hard to "game" the system (i.e., to find a way to change machine output so that BLEU goes up, but quality doesn't)

BLEU Evaluation Metric

(Papineni et al, ACL-2002)

Reference (human) translation:

The U.S. island of Guam is maintaining a high state of alert after the Guam airport and its offices both received an e-mail from someone calling himself the Saudi Arabian Osama bin Laden and threatening a biological/chemical attack against public places such as the airport .

Machine translation:

The American [?] international airport and its the office all receives one calls self the sand Arab rich business [?] and so on electronic mail , which sends out ; The threat will be able after public place and so on the airport to start the biochemistry attack , [?] highly alerts after the maintenance.

- BLEU is a weighted geometric mean, with a brevity penalty factor added.
  - Note that it's precision-oriented
- BLEU4 formula
  (counts n-grams up to length 4)

exp (0.5 * log p1 +
0.25 * log p2 +
0.125 * log p3 +
0.125 * log p4 –
max(words-in-reference / words-in-machine – 1, 0)

p1 = 1-gram precision
P2 = 2-gram precision
P3 = 3-gram precision
P4 = 4-gram precision

Note: only works at corpus level (zeroes kill it); there's a smoothed variant for sentence-level

BLEU in Action

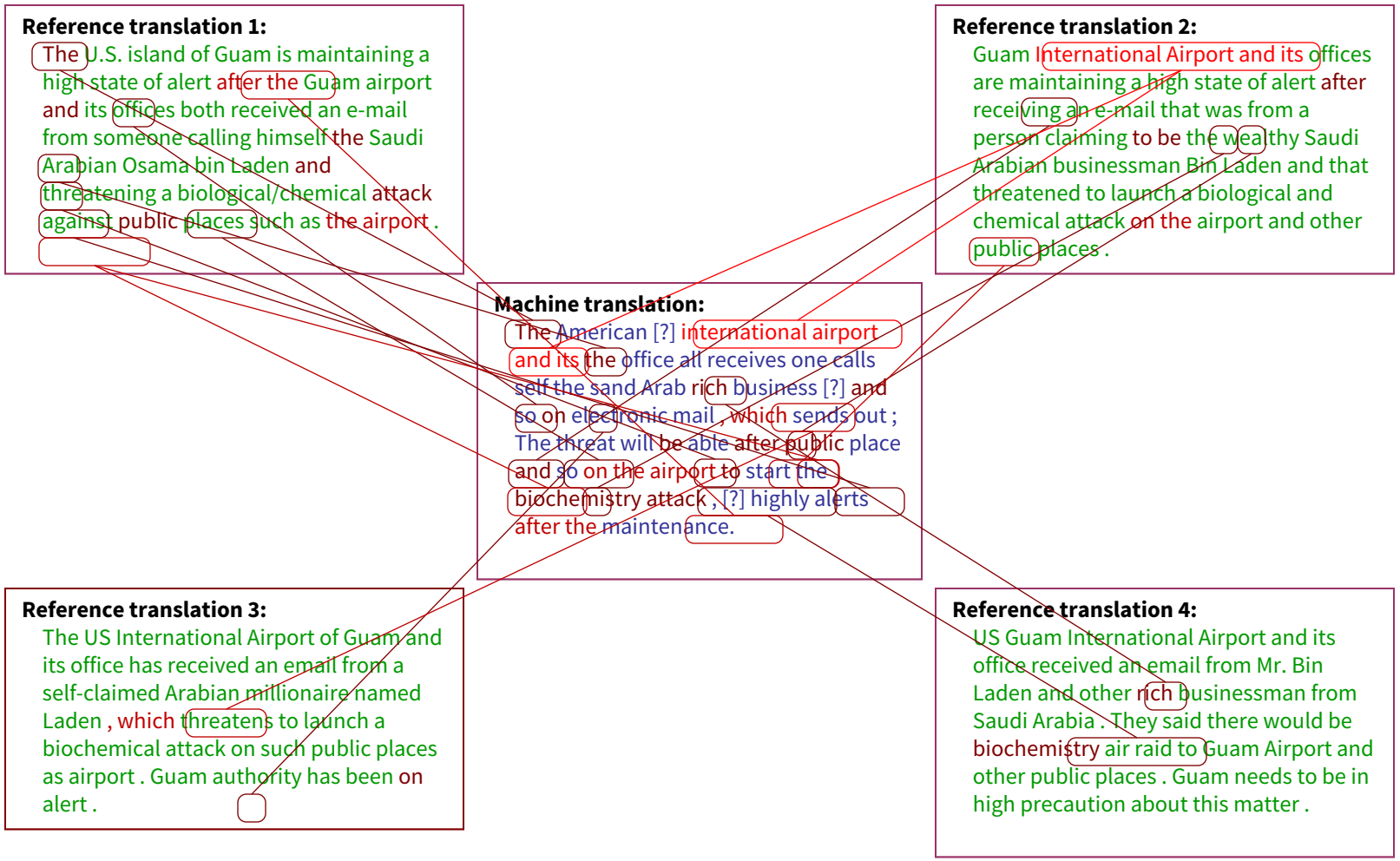
枪手被警方击毙。 (Foreign Original)

the gunman was shot to death by the police . (Reference Translation)

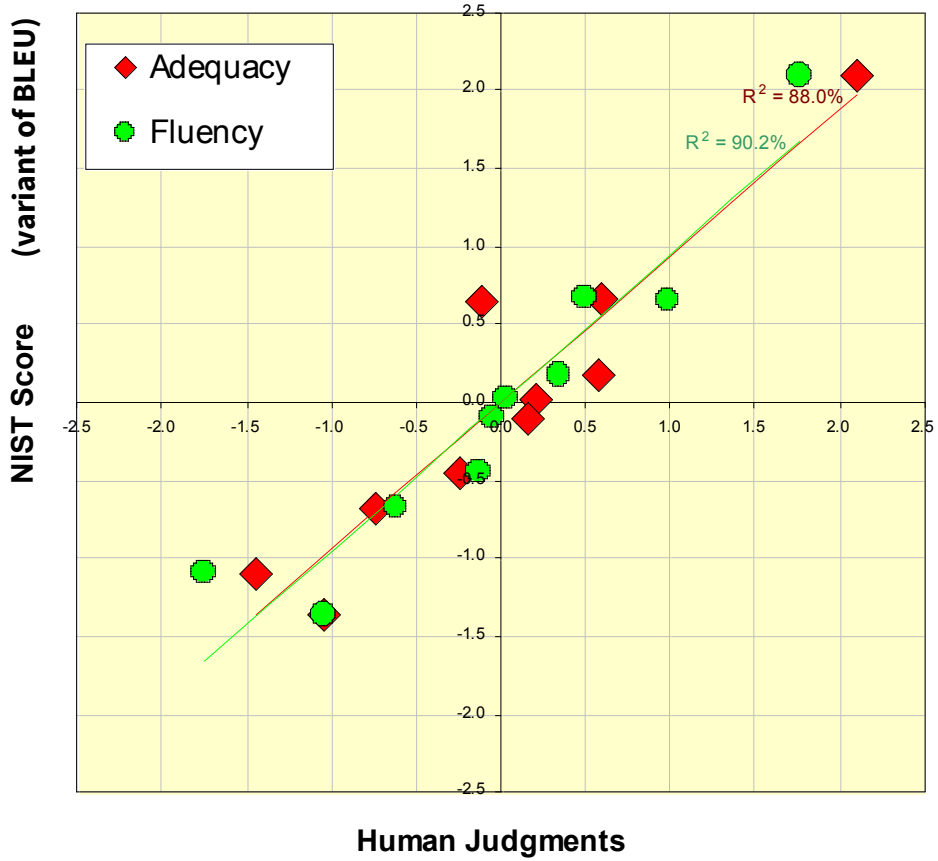
the gunman was police kill .	#1
wounded police jaya of	#2
the gunman was shot dead by the police .	#3
the gunman arrested by police kill .	#4
the gunmen were killed .	#5
the gunman was shot to death by the police .	#6
gunmen were killed by police ?SUB>0 ?SUB>0	#7
al by the police .	#8
the ringer is killed by the police .	#9
police killed the gunman .	#10

green	= 4-gram match	(good!)
red	= word not matched	(bad!)

Multiple Reference Translations



Initial results showed that BLEU predicts human judgments well



slide from G. Doddington (NIST)

Automatic evaluation of MT

- People started optimizing their systems to maximize BLEU score
  - BLEU scores improved rapidly
  - The correlation between BLEU and human judgments of quality went way, way down
  - MT BLEU scores now approach those of human translations but their true quality remains far below human translations
- Coming up with automatic MT evaluations has become its own research field
  - There are many proposals: TER, METEOR, MaxSim, SEPIA, our own RTE-MT
  - TERpA is a representative good one that handles some word choice variation.
- MT research **requires** *some* automatic metric to allow a rapid development and evaluation cycle.

6. Doing your research example: Straightforward Class Project: Apply NNets to Task

1. Define Task:
   - Example: **Summarization**

2. Define Dataset
   1. Search for academic datasets
      - They already have baselines
      - E.g.: Newsroom Summarization Dataset: https://summari.es

   2. Define your own data (harder, need new baselines)
      - Allows connection to your research
      - A fresh problem provides fresh opportunities!
      - Be creative: Twitter, Blogs, News, etc. There are lots of neat websites which provide creative opportunities for new tasks

Straightforward Class Project: Apply NNets to Task

3. Dataset hygiene
   - Right at the beginning, separate off devtest and test splits
     - Discussed more next

4. Define your metric(s)
   - Search online for well established metrics on this task
   - Summarization: Rouge (Recall-Oriented Understudy for Gisting Evaluation) which defines *n*-gram overlap to human summaries
   - Human evaluation is still much better for summarization; you may be able to do a small scale human eval

Straightforward Class Project: Apply NNets to Task

5. Establish a baseline
   - Implement the simplest model first (often logistic regression on unigrams and bigrams or averaging word vectors)
     - For summarization: See LEAD-3 baseline
   - Compute metrics on train AND dev
   - Analyze errors
   - If metrics are amazing and no errors:
     - Done! Problem was too easy. Need to restart. ☺/☹

6. Implement existing neural net model
   - Compute metric on train and dev
   - Analyze output and errors
   - Minimum bar for this class

Straightforward Class Project: Apply NNets to Task

7. Always be close to your data! (Except for the final test set!)
   - Visualize the dataset
   - Collect summary statistics
   - Look at errors
   - Analyze how different hyperparameters affect performance

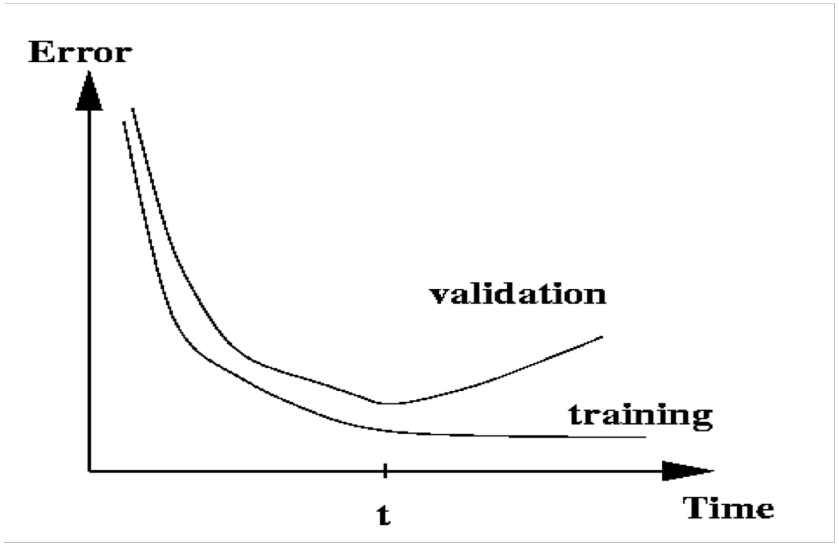
8. Try out different models and model variants
   Aim to iterate quickly via having a good experimental setup
   - Fixed window neural model
   - Recurrent neural network
   - Recursive neural network
   - Convolutional neural network
   - Attention-based model
   - …

Pots of data

- 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, though you have to weigh its size/usefulness against the reduction in train-set size.
- Having a fixed test set ensures that all systems are assessed against the same gold data. This is generally good, but it is problematic where 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

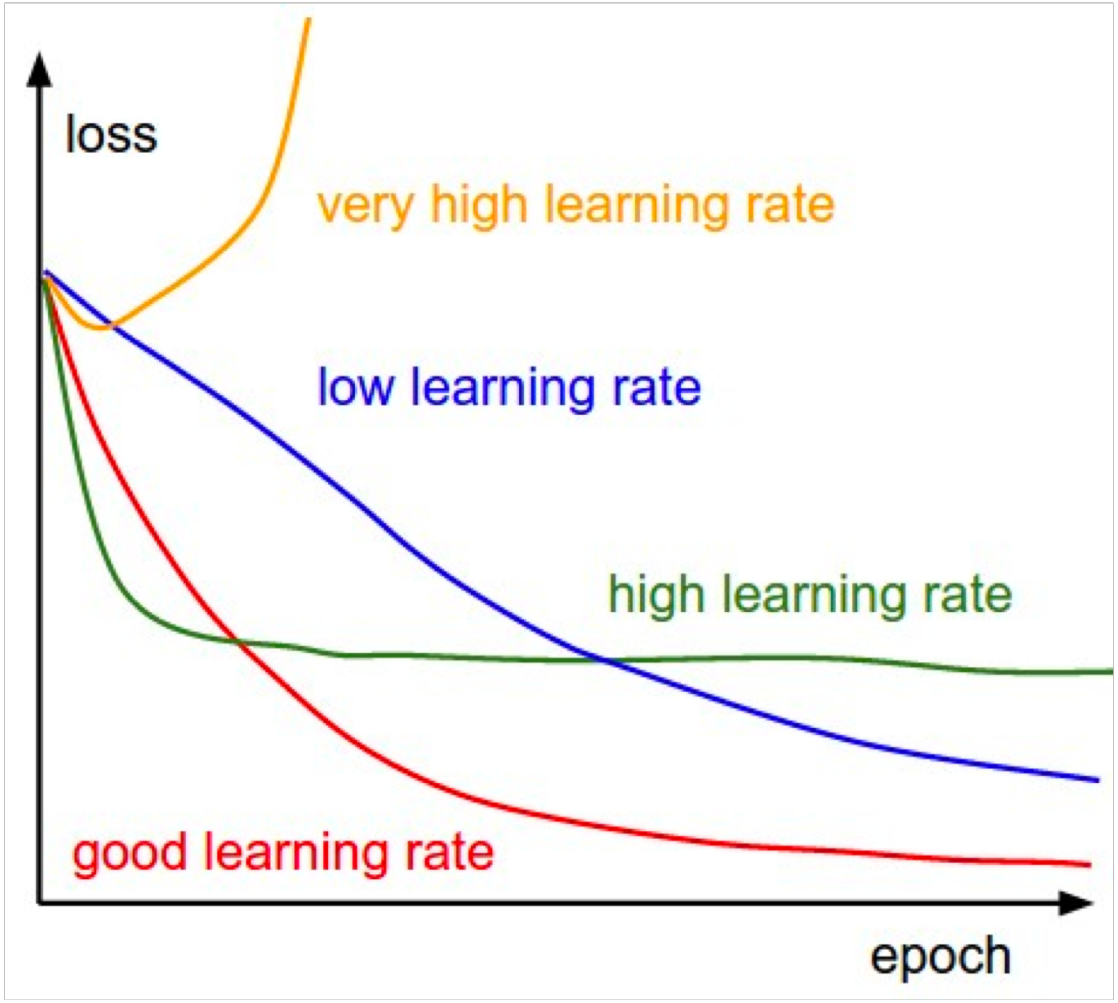
- The **train**, **tune**, **dev**, and **test** sets need to be completely distinct
- It is invalid to test on material you have trained on
  - You will get a falsely good performance. We usually overfit on train
- You need an independent tuning set
  - The hyperparameters won't be set right if tune 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!

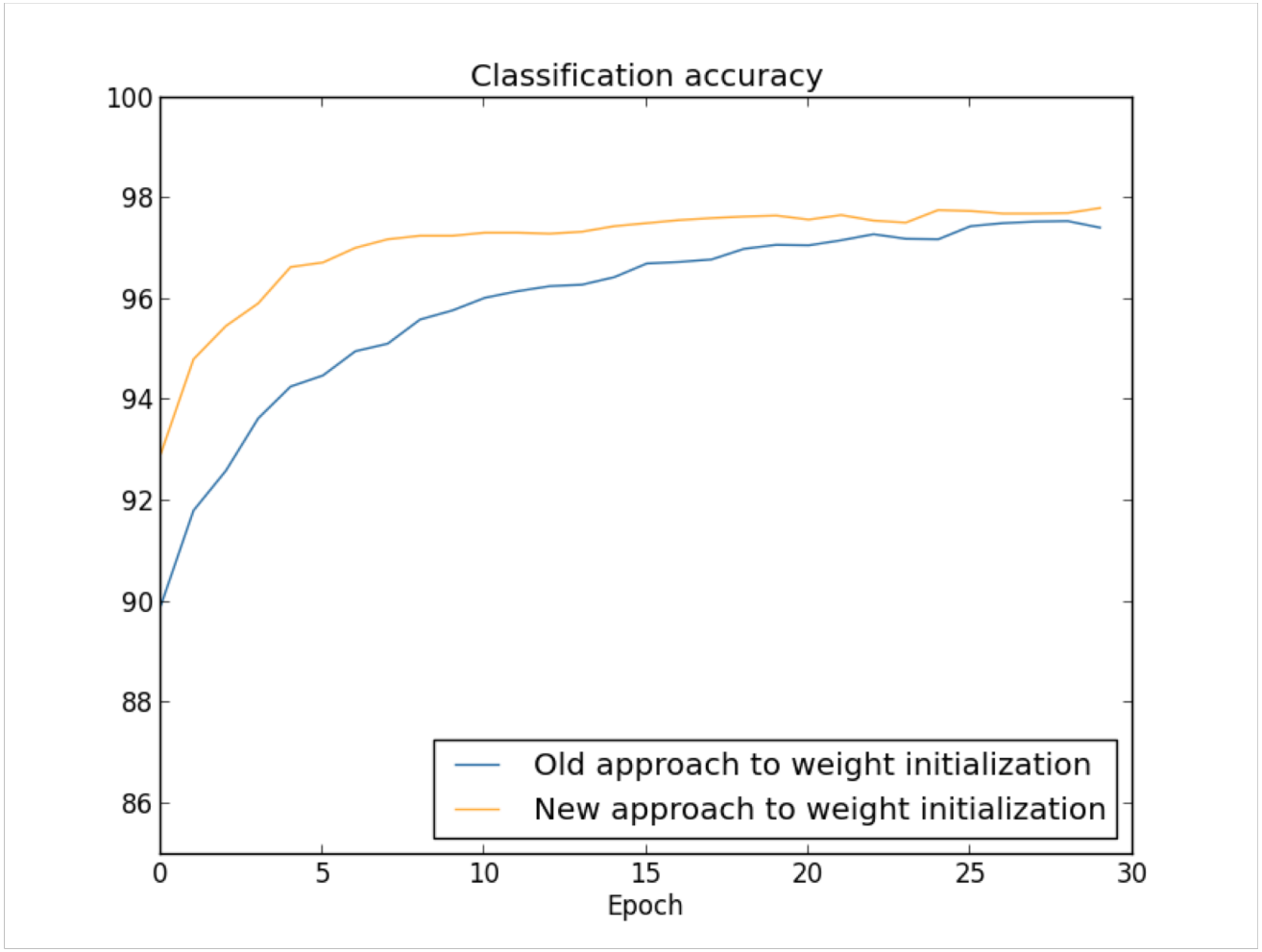
Models are sensitive to learning rates

- From Andrej Karpathy, CS231n course notes



Models are sensitive to initialization

- From Michael Nielsen
  http://neuralnetworksanddeeplearning.com/chap3.html



Training a (gated) RNN

1. Use an LSTM or GRU: *it makes your life so much simpler!*
2. Initialize recurrent matrices to be orthogonal
3. Initialize other matrices with a sensible (**small!**) scale
4. Initialize forget gate bias to 1: *default to remembering*
5. Use adaptive learning rate algorithms: *Adam, AdaDelta, …*
6. Clip the norm of the gradient: *1–5 seems to be a reasonable threshold when used together with Adam or AdaDelta.*
7. Either only dropout vertically or look into using Bayesian Dropout (Gal and Gahramani – not natively in PyTorch)
8. *Be patient! Optimization takes time*

[Saxe et al., ICLR2014;
Ba, Kingma, ICLR2015;
Zeiler, arXiv2012;
Pascanu et al., ICML2013]

Experimental strategy

- Work incrementally!
- Start with a very simple model and get it to work
- 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
  - Something like 8 examples is good
  - Often synthetic data is useful for this
  - Make sure you can get 100% on this data
    - Otherwise your model is definitely either not powerful enough or it is broken

Experimental strategy

- 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 be scared of 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 is **really** important to almost all of the successes of NNets

Project writeup

- Writeup quality is important
  - Look at last-year's prize winners for examples

Good luck with your projects!