Recurrent Neural Networks for text analysis

From idea to practice

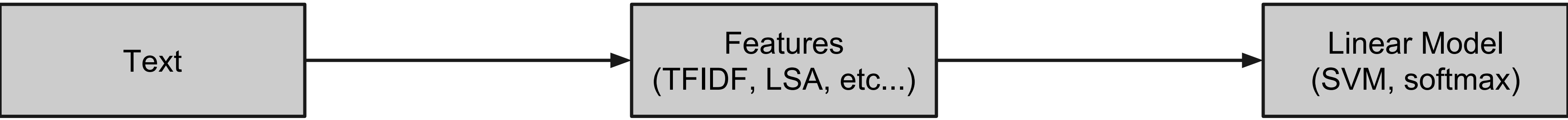
Follow Along

Slides at: http://goo.gl/6f8cnc

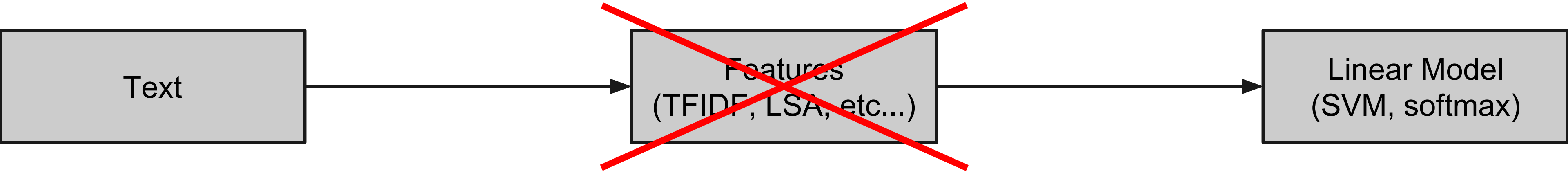
How ML

-0.15, 0.2, 0, 1.5	Numerical, great!
A, B, C, D	Categorical, great!
The cat sat on the mat.	Uhhh…….

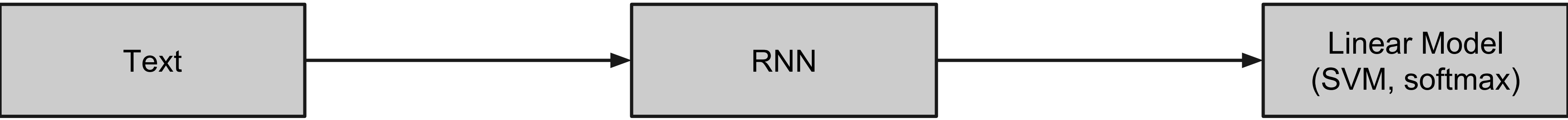
How text is dealt with (ML perspective)



How text is dealt with (ML perspective)

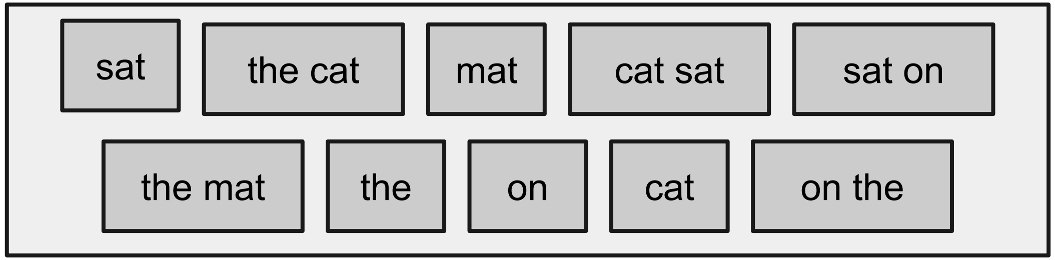


How text should be dealt with?



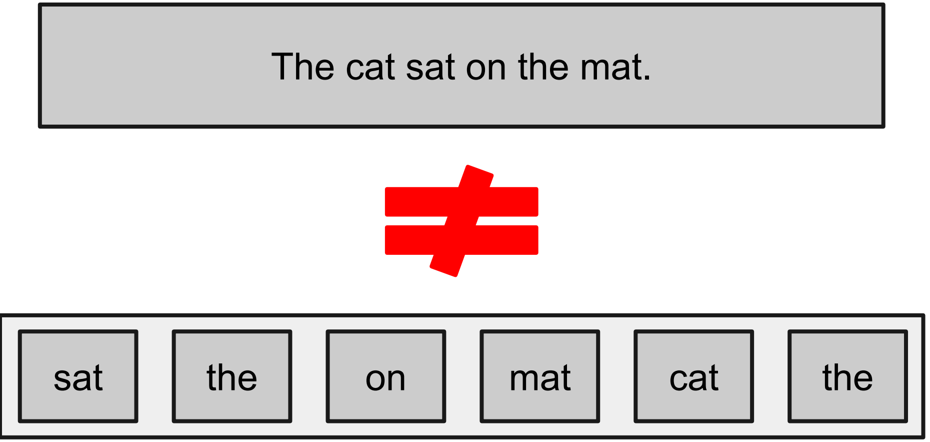
Structure is hard

Ngrams is typical way of preserving some structure.



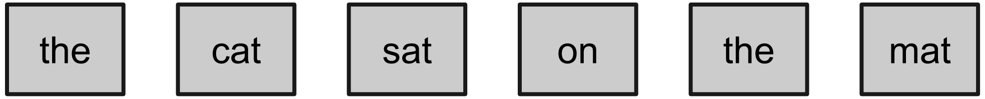
Beyond bi or tri-grams occurrences become very rare and dimensionality becomes huge (1, 10 million + features)

Structure is important!

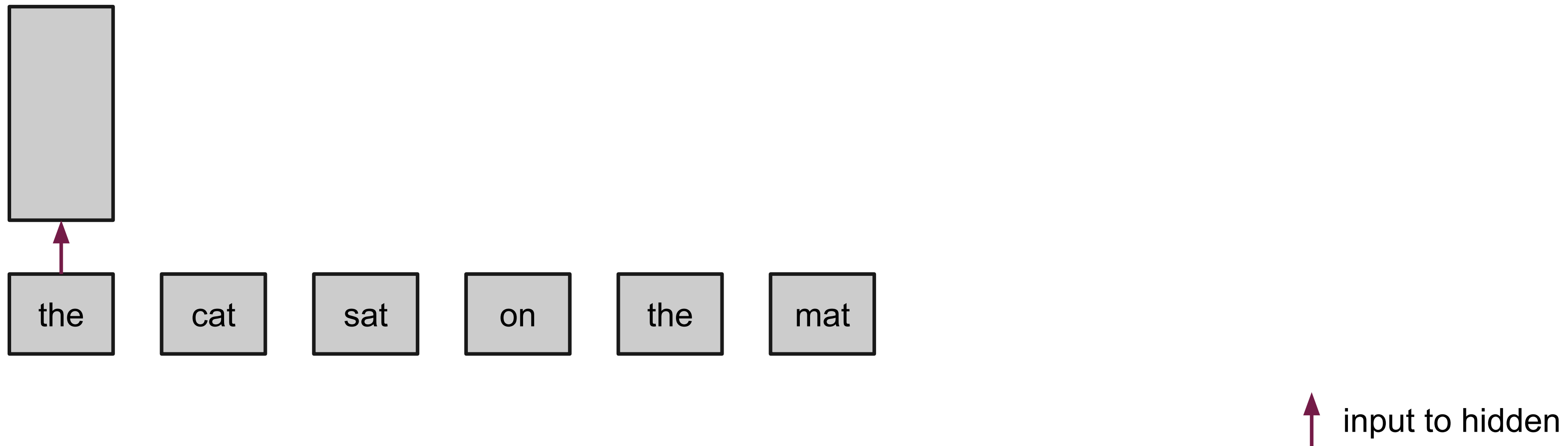


- Certain tasks, structure is essential:
  - Humor
  - Sarcasm
- Certain tasks, ngrams can get you a long way:
  - Sentiment Analysis
  - Topic detection
- Specific words can be strong indicators
  - useless, fantastic (sentiment)
  - hoop, green tea, NASDAQ (topic)

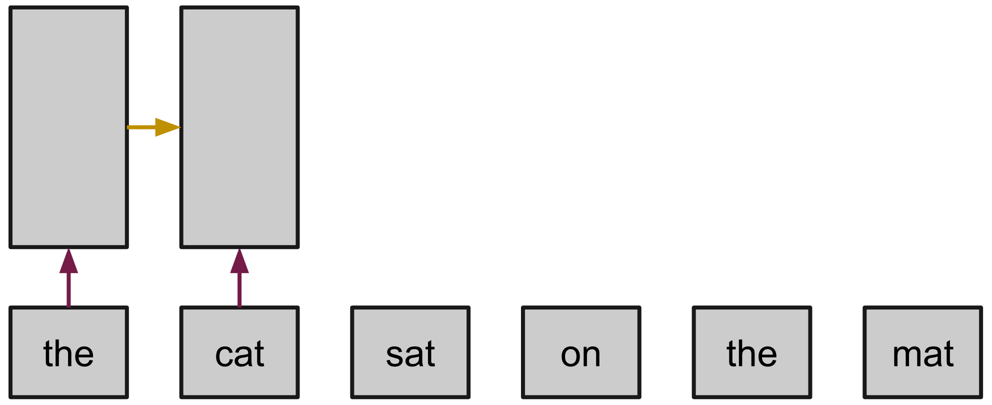
How an RNN works

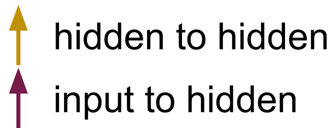


How an RNN works

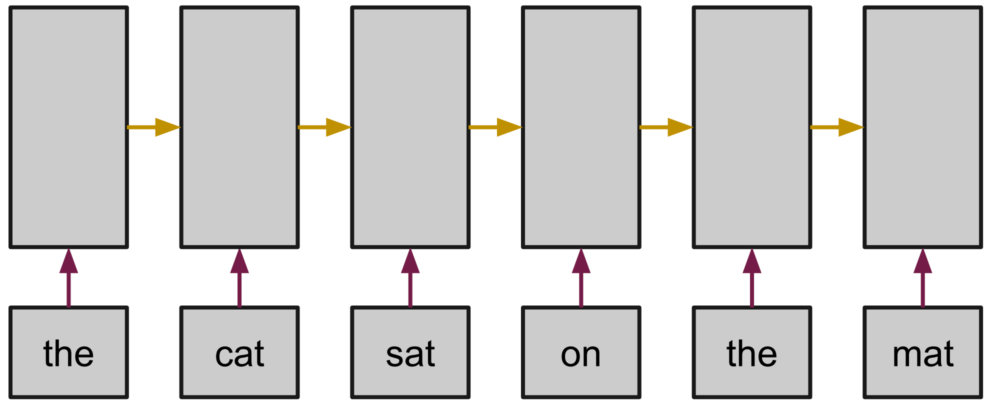


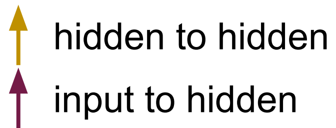
How an RNN works



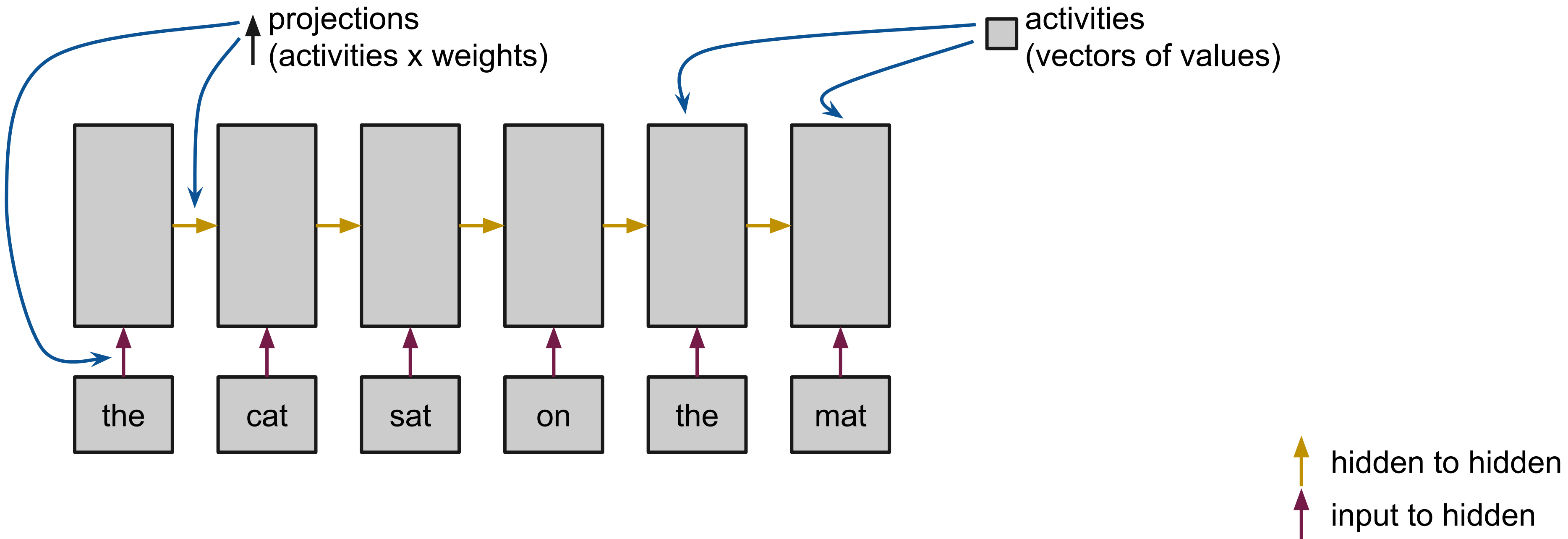


How an RNN works

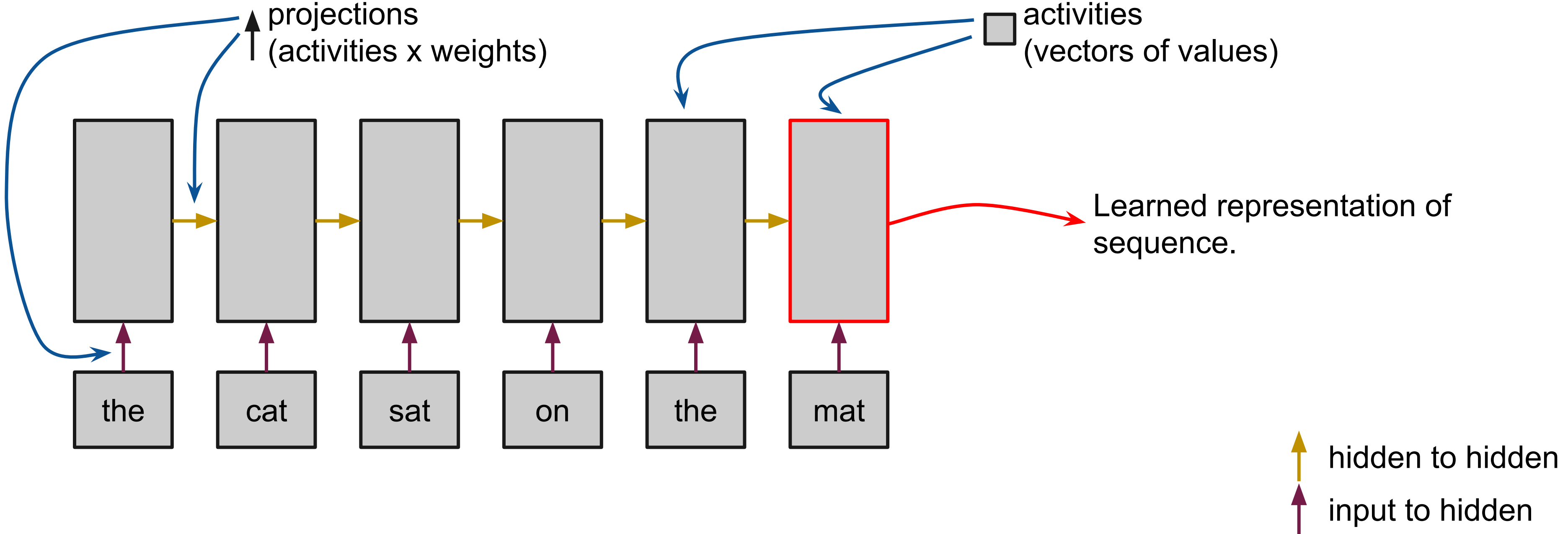




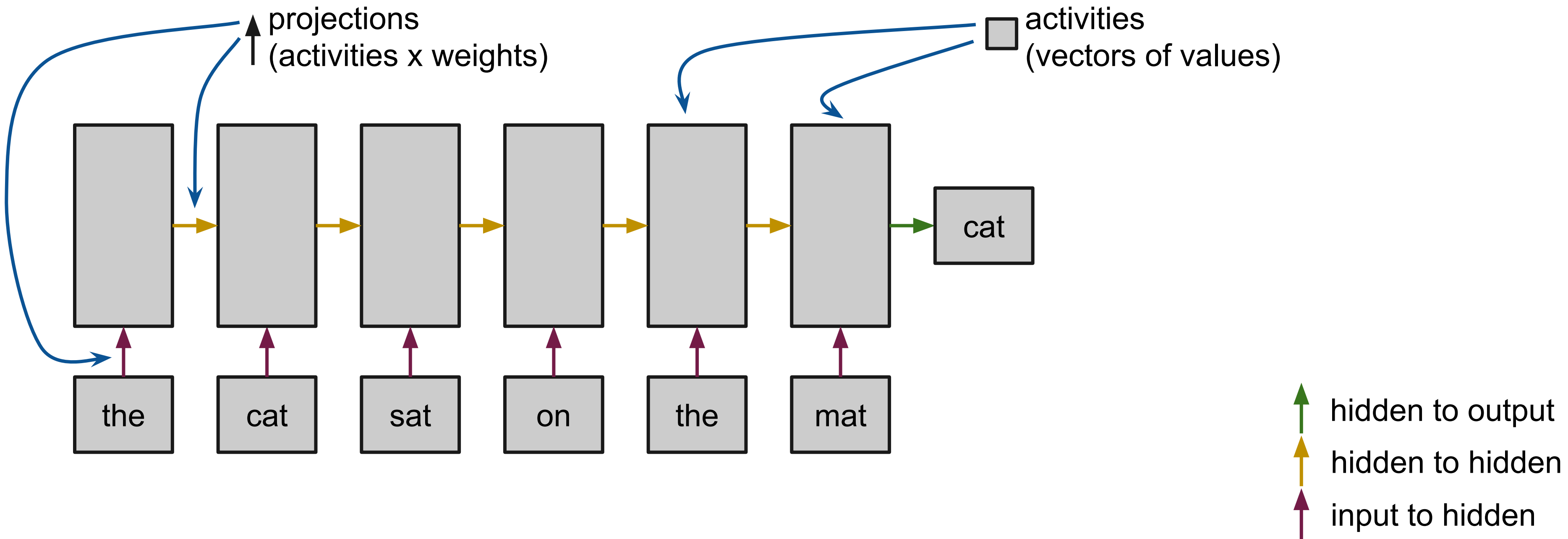
How an RNN works



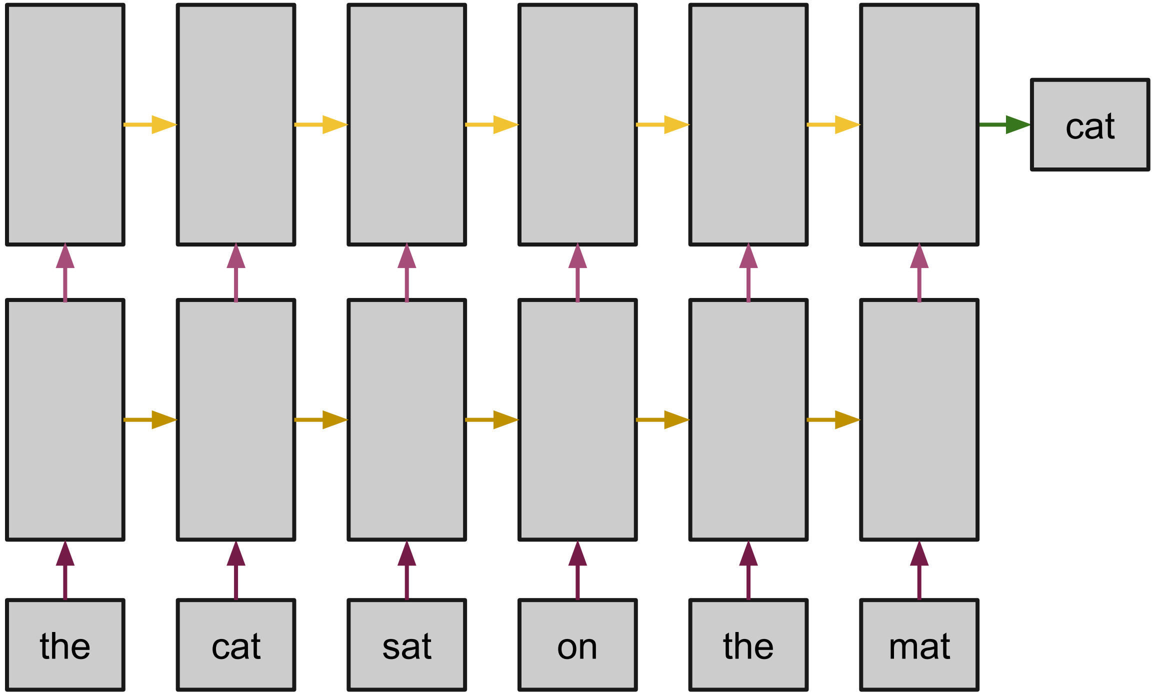
How an RNN works

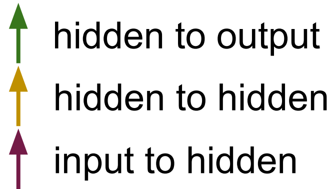


How an RNN works



You can stack them too





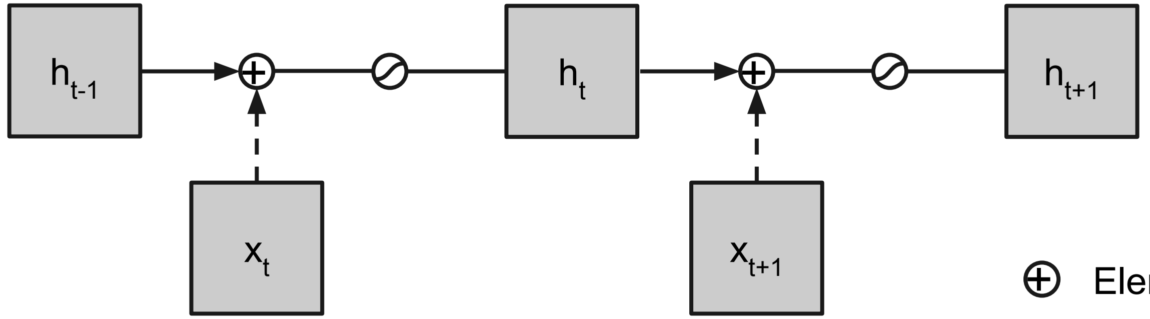
But aren’t RNNs unstable?

Simple RNNs trained with SGD are unstable/difficult to learn.

But modern RNNs with various tricks blow up much less often!

- Gating Units
- Gradient Clipping
- Steeper gates
- Orthogonal initialization
- Better optimizers
- Bigger datasets

Simple Recurrent Unit



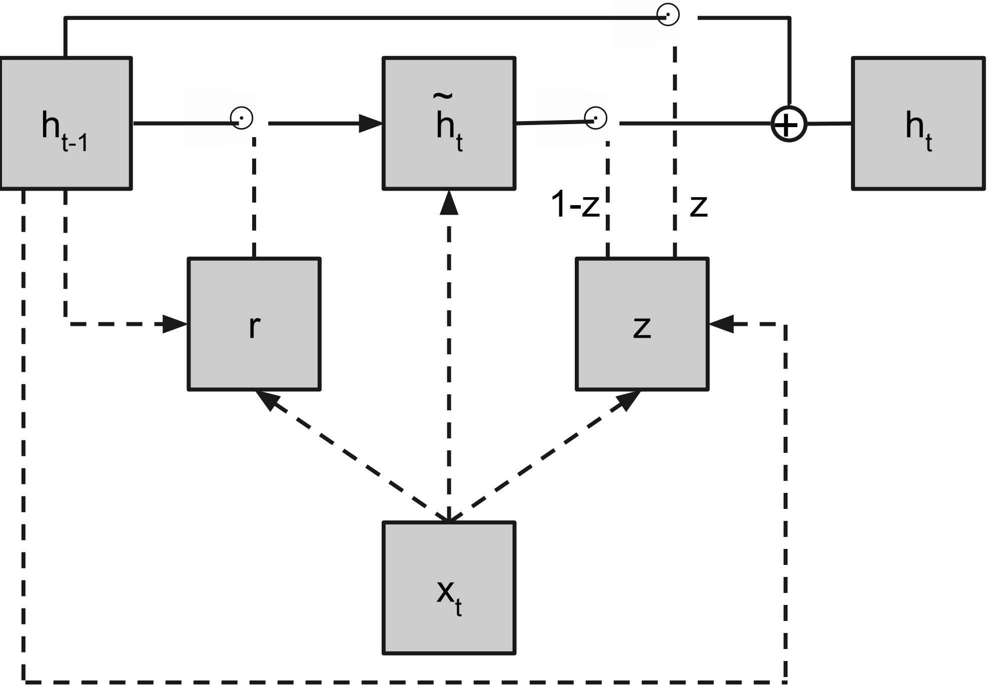
⊕ Element wise addition

⊘ Activation function

↑ Routes information can propagate along

⇡ Involved in modifying information flow and values

Gated Recurrent Unit - GRU



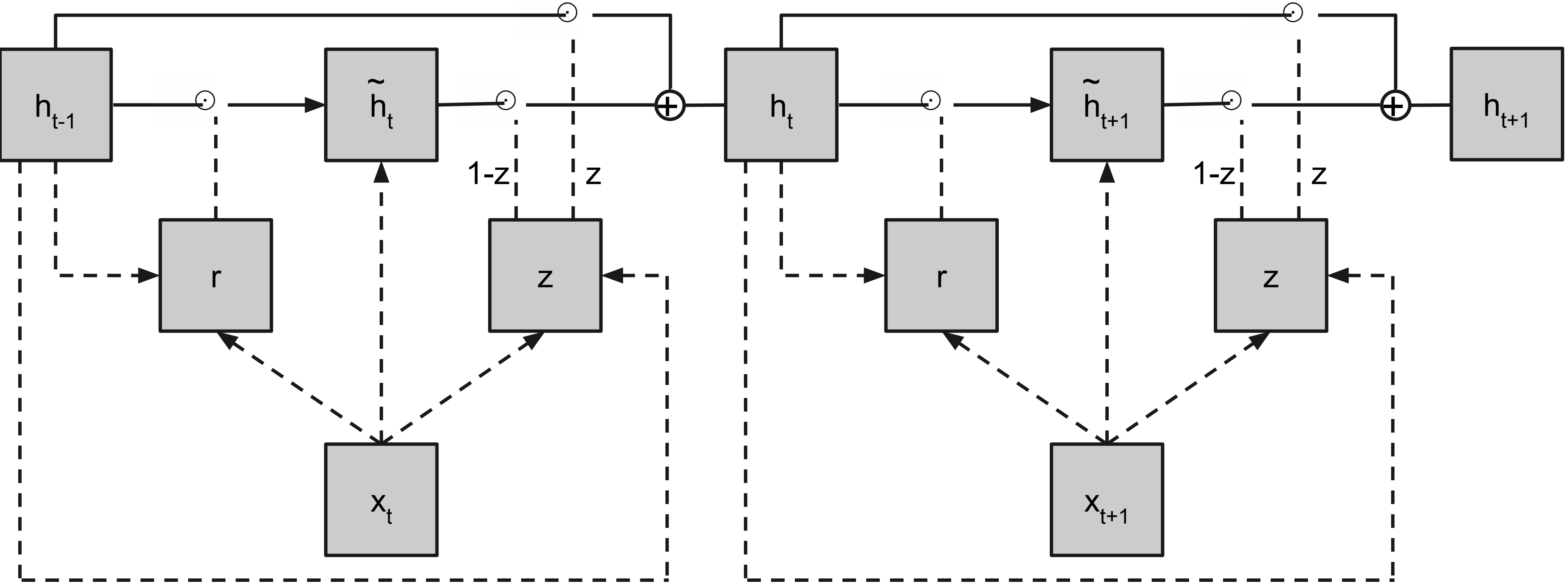
⊕ Element wise addition

⊙ Element wise multiplication

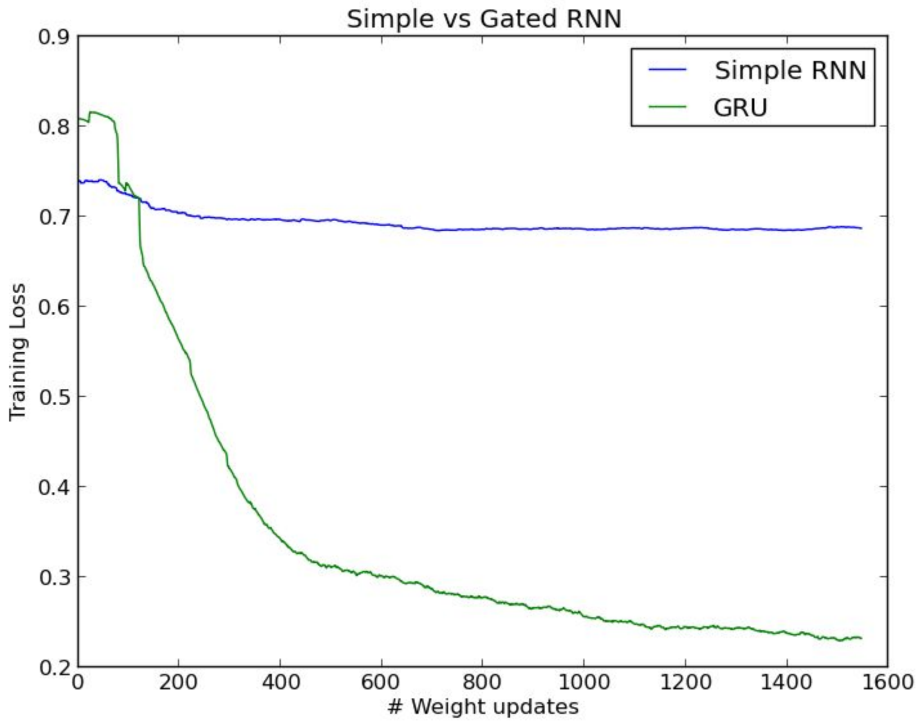
↑ Routes information can propagate along

⇡ Involved in modifying information flow and values

Gated Recurrent Unit - GRU



Gating is essential



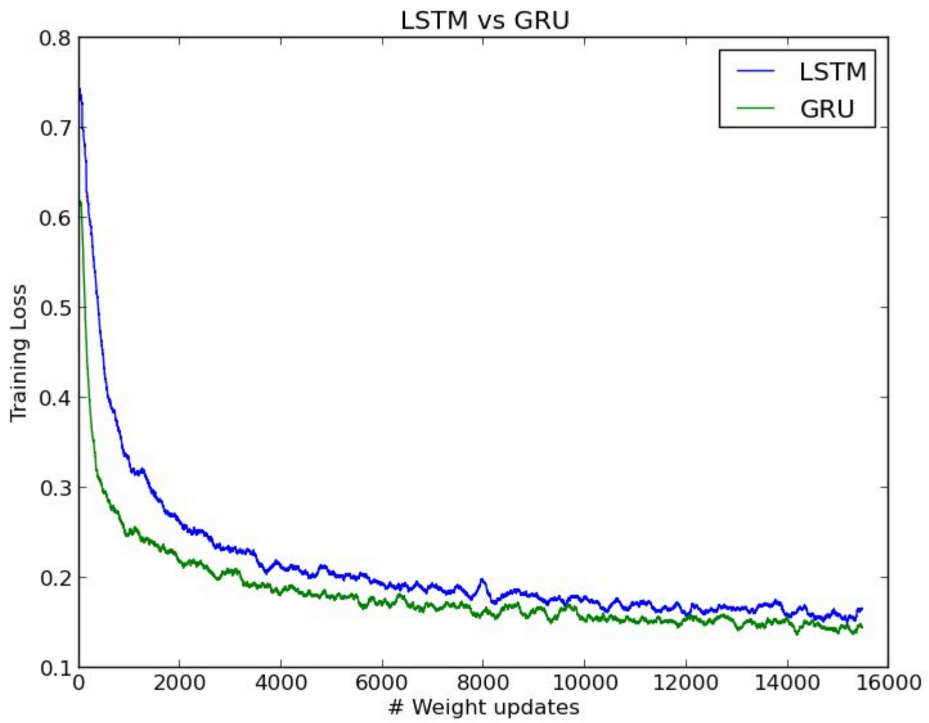
For sentiment analysis of longer sequences of text (paragraph or so) a simple RNN has difficulty learning at all while a gated RNN does so easily.

Which One?

There are two types of gated RNNs:

- Gated Recurrent Units (GRU) by K. Cho, recently introduced and used for machine translation and speech recognition tasks.

- Long short term memory (LSTM) by S. Hochreiter and J. Schmidhuber has been around since 1997 and has been used far more. Various modifications to it exist.

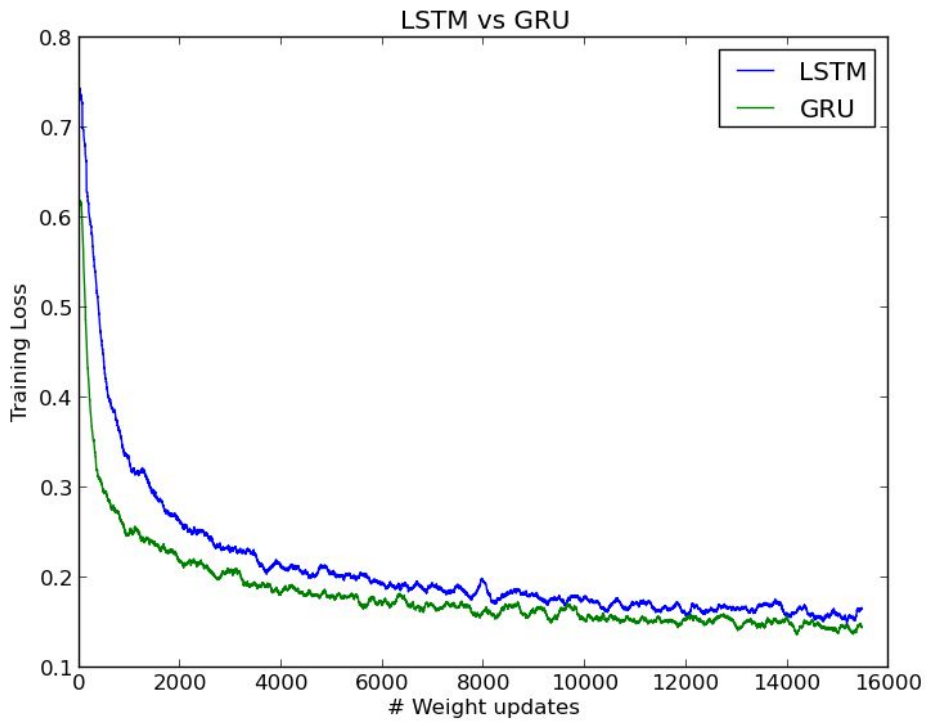


Which One?

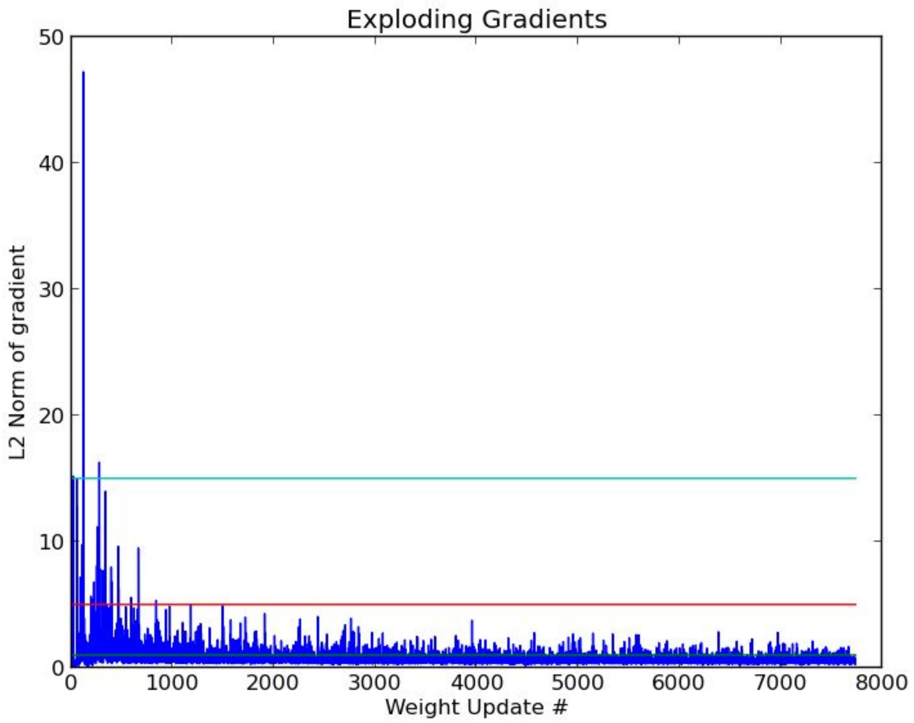
GRU is simpler, faster, and optimizes quicker (at least on sentiment).

Because it only has two gates (compared to four) approximately 1.5-1.75x faster for theano implementation.

If you have a huge dataset and don’t mind waiting LSTM may be better in the long run due to its greater complexity - especially if you add peephole connections.



Exploding Gradients?



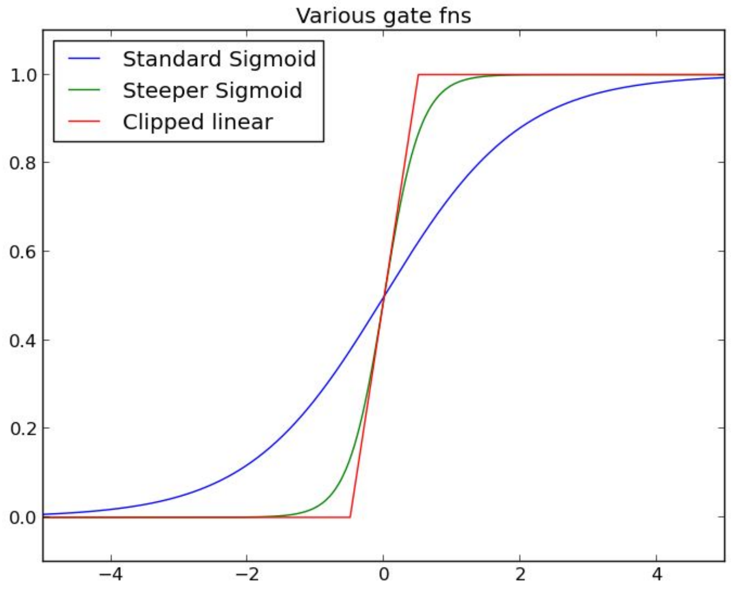
Exploding gradients are a major problem for traditional RNNs trained with SGD. One of the sources of the reputation of RNNs being hard to train.

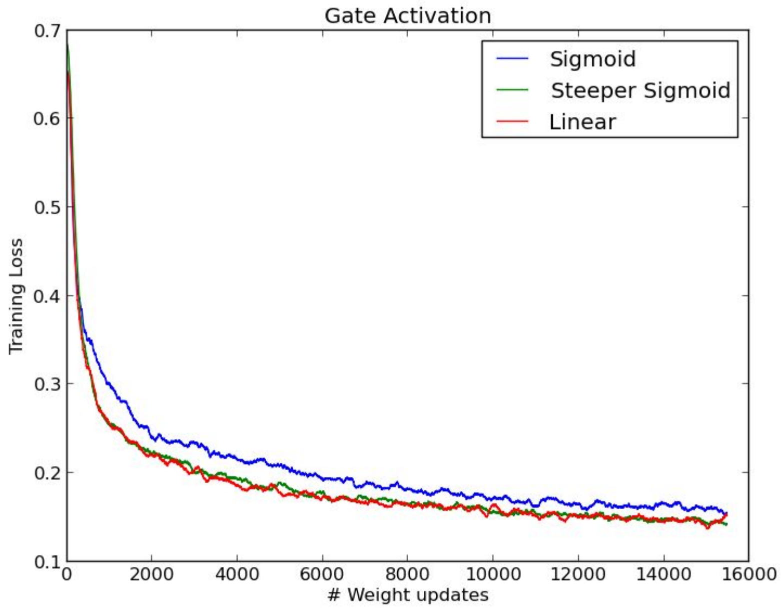
In 2012, R Pascanu and T. Mikolov proposed clipping the norm of the gradient to alleviate this.

Modern optimizers don’t seem to have this problem - at least for classification text analysis.

Better Gating Functions

Interesting paper at NIPS workshop (Q. Lyu, J. Zhu) - make the gates “steeper” so they change more rapidly from “off” to “on” so model learns to use them quicker.



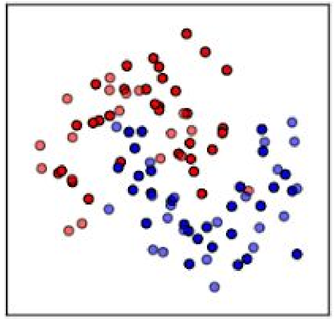


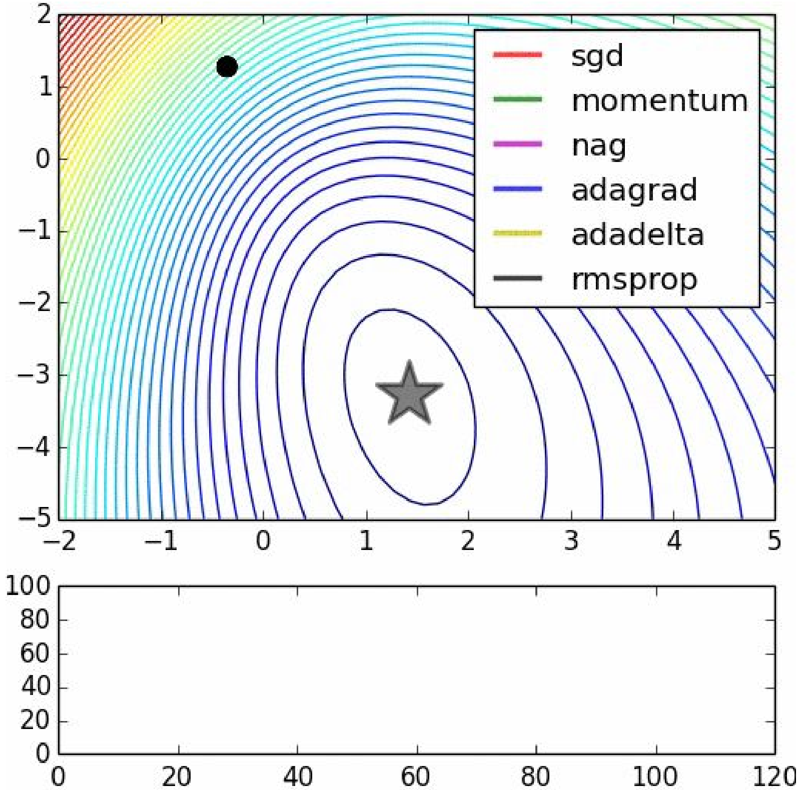
Orthogonal Initialization

Andrew Saxe last year did analysis of deep linear networks showing that initializing weight matrices with random orthogonal matrices works better than random gaussian (or uniform) matrices.

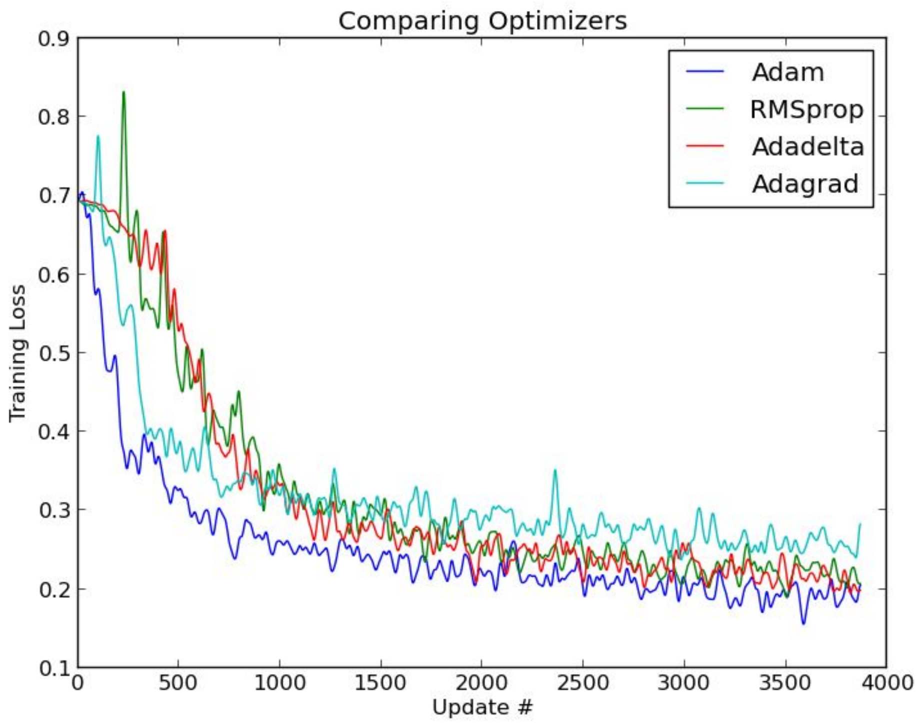
Various previous work also noted various similar forms of initialization worked well for RNNs.

Understanding Optimizers


2D moons dataset
courtesy of scikit-learn



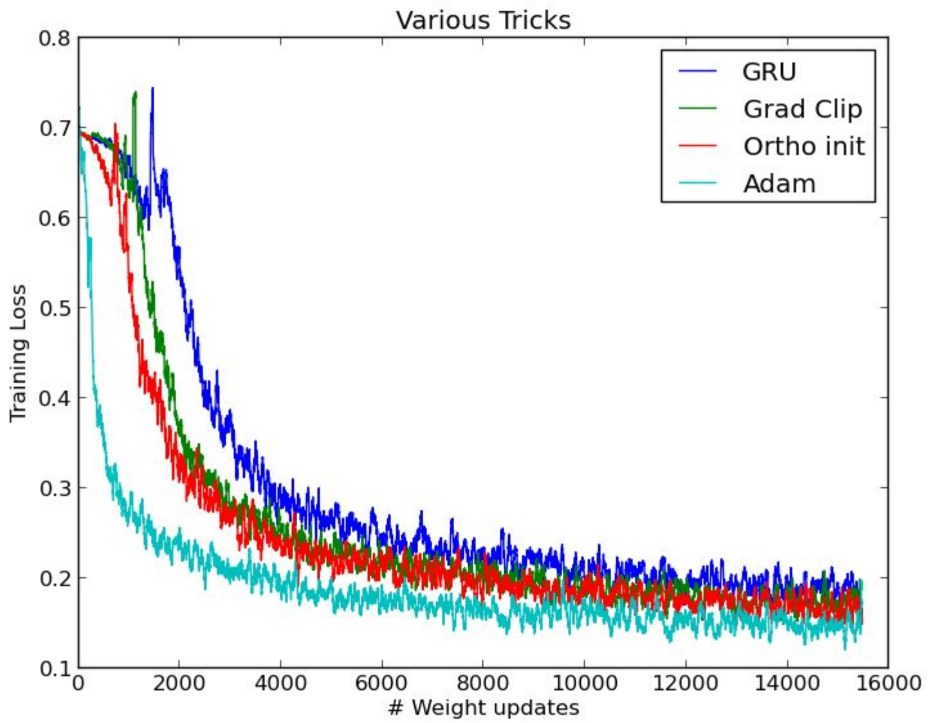
Comparing Optimizers



Adam (D. Kingma) combines the early optimization speed of Adagrad (J. Duchi) with the better later convergence of various other methods like Adadelta (M. Zeiler) and RMSprop (T. Tieleman).

Warning: Generalization performance of Adam seems slightly worse for smaller datasets.

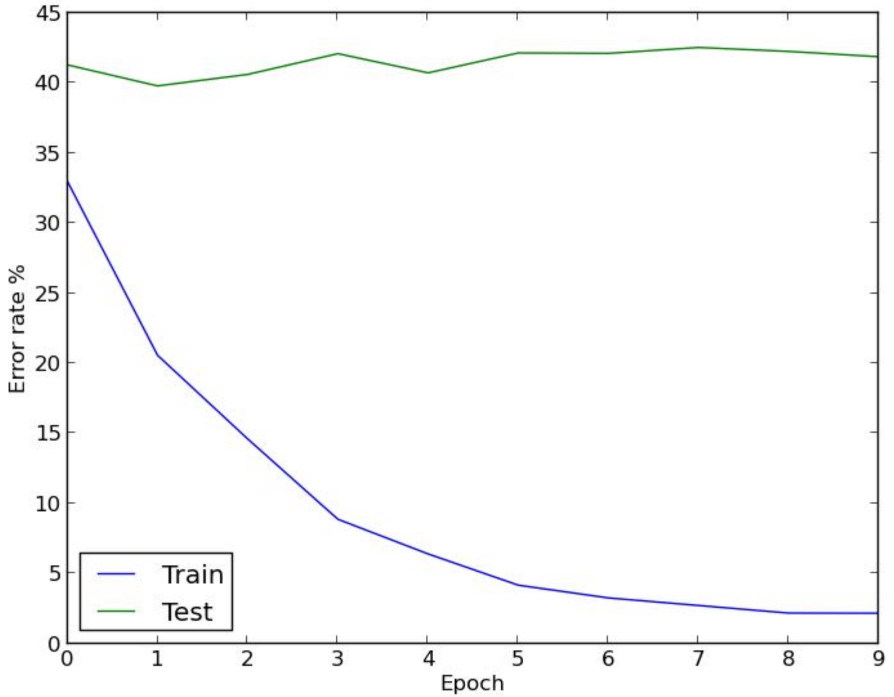
It adds up



Up to 10x more efficient training once you add all the tricks together compared to a naive implementation - much more stable - rarely diverges.

Around 7.5x faster, the various tricks add a bit of computation time.

Too much? - Overfitting

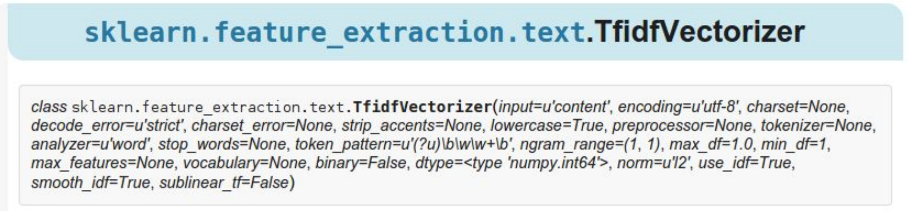


RNNs can overfit very well as we will see. As they continue to fit to training dataset, their performance on test data will plateau, or even worsen.

Keep track of it using a validation set, save model at each iteration over training data and pick the earliest, best, validation performance.

The Showdown

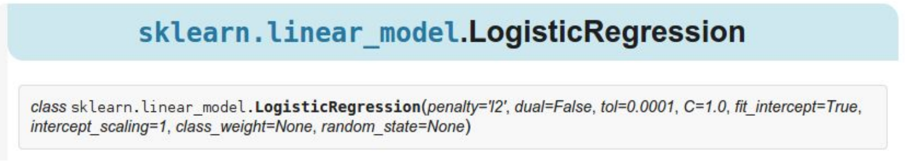
Model #1



sklearn.feature_extraction.text.TfidfVectorizer

class `sklearn.feature_extraction.text.`**`TfidfVectorizer`**(*input=u'content', encoding=u'utf-8', charset=None, decode_error=u'strict', charset_error=None, strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, analyzer=u'word', stop_words=None, token_pattern=u'(?u)\b\w\w+\b', ngram_range=(1, 1), max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<type 'numpy.int64'>, norm=u'l2', use_idf=True, smooth_idf=True, sublinear_tf=False*)

+

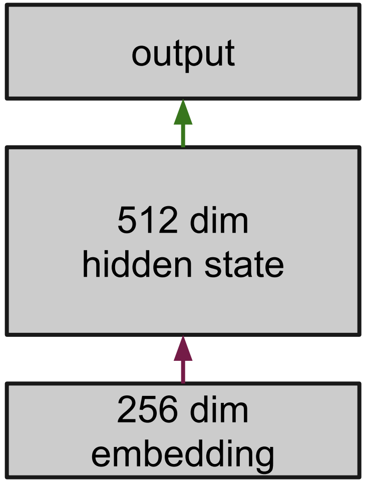


sklearn.linear_model.LogisticRegression

class `sklearn.linear_model.`**`LogisticRegression`**(*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None*)

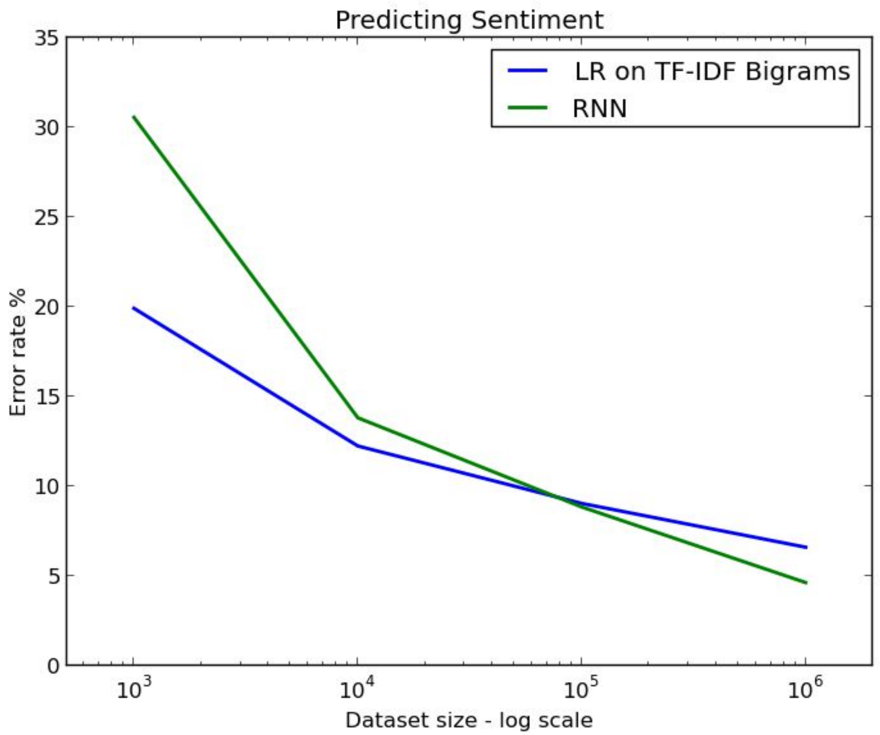
Using bigrams and grid search on min_df for vectorizer and regularization coefficient for model.

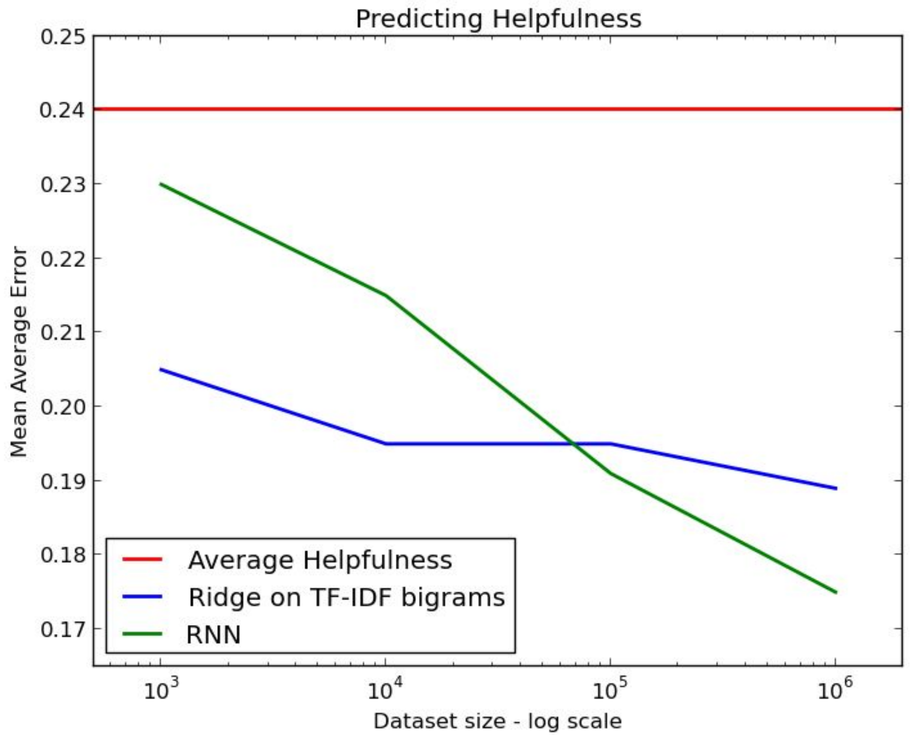
Model #2



Using whatever I tried that worked :)
Adam, GRU, steeper sigmoid gates, ortho init are good defaults

Sentiment & Helpfulness





Effect of Dataset Size

- RNNs have poor generalization properties on small datasets.
  - 1K labeled examples 25-50% worse than linear model…
- RNNs have better generalization properties on large datasets.
  - 1M labeled examples 0-30% better than linear model.
- Cross over appears to be between 100K and 1M examples
  - Depends on dataset.

The Thing we don’t talk about

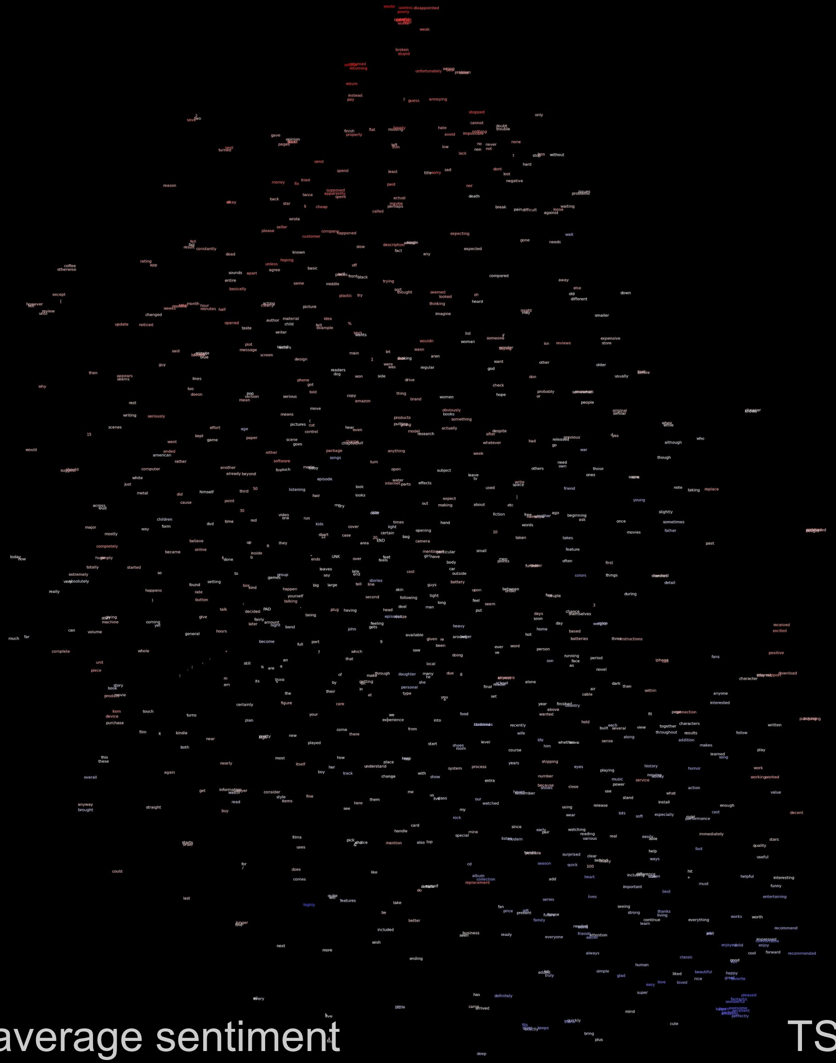
For 1 million paragraph sized text examples to converge:

- Linear model takes 30 minutes on a single CPU core.
- RNN takes 2 hours on a GTX 980.
- RNN takes five days on a single CPU core.

RNN is about 250x slower on CPU than linear model…

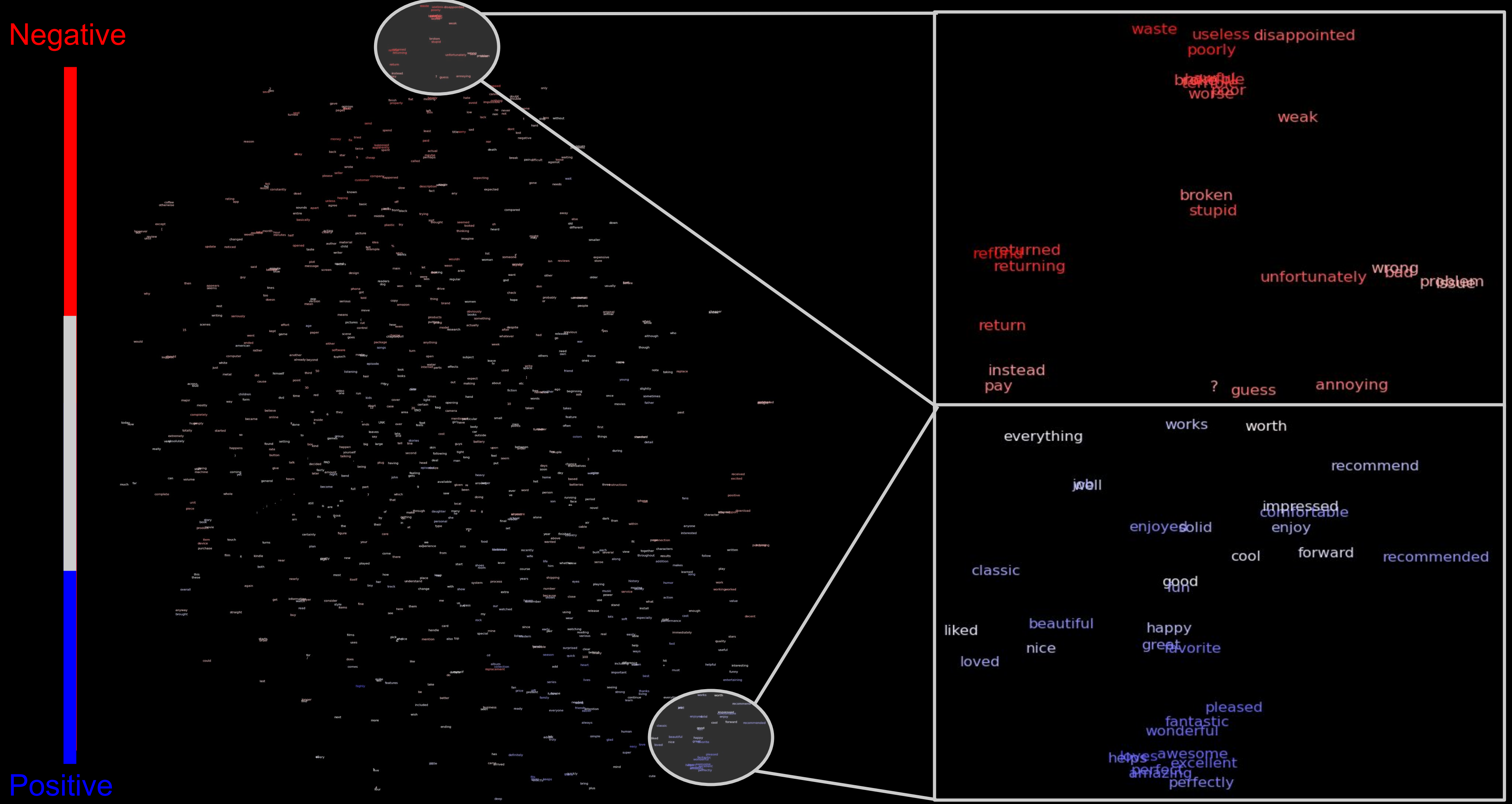
This is why we use GPUs.

Visualizing representations of words learned via sentiment

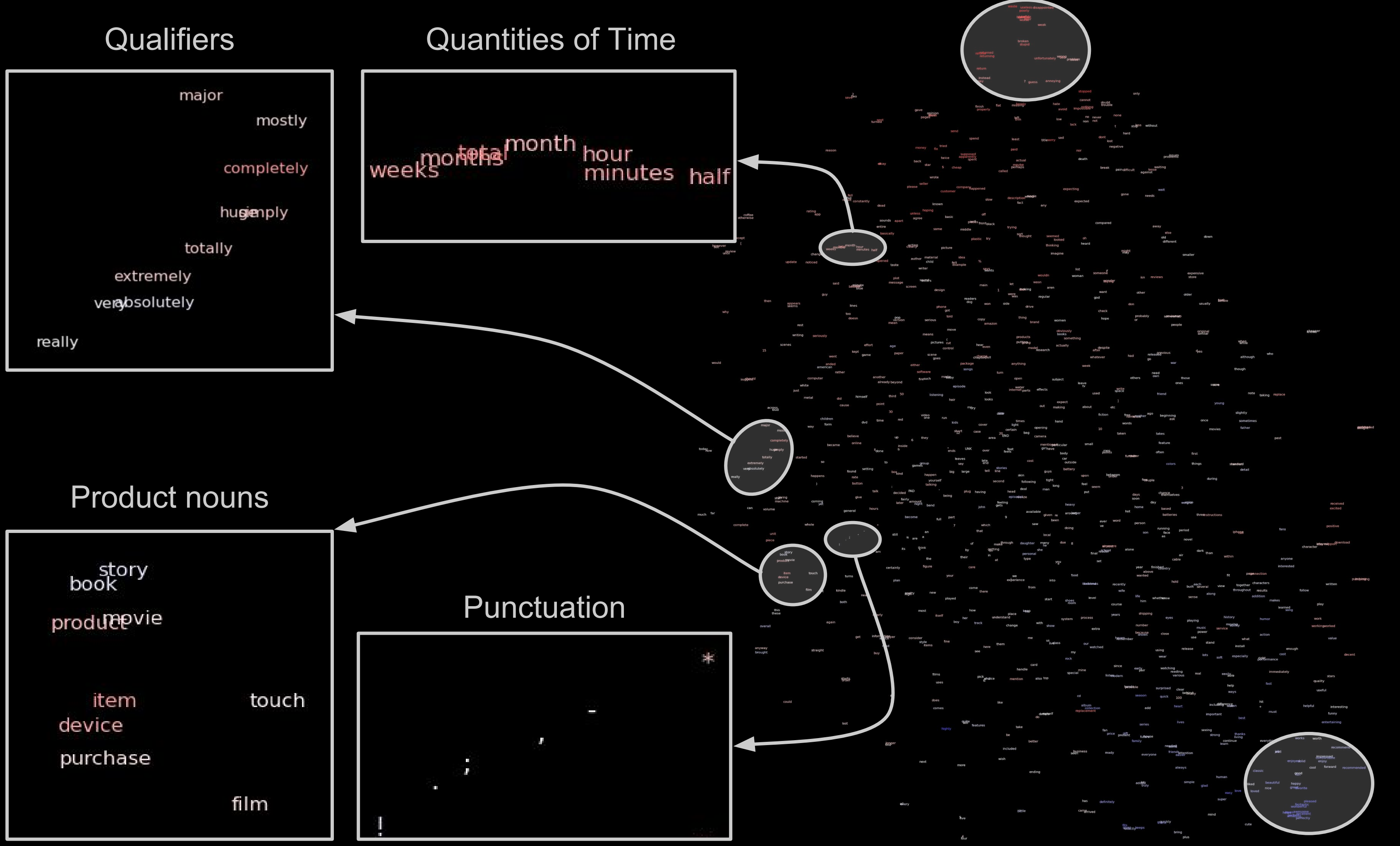


Individual words colored by average sentiment

TSNE - L.J.P. van der



Model learns to separate negative and positive words, not too surprising



Much cooler, model also begins to learn components of language ***from only binary sentiment labels***

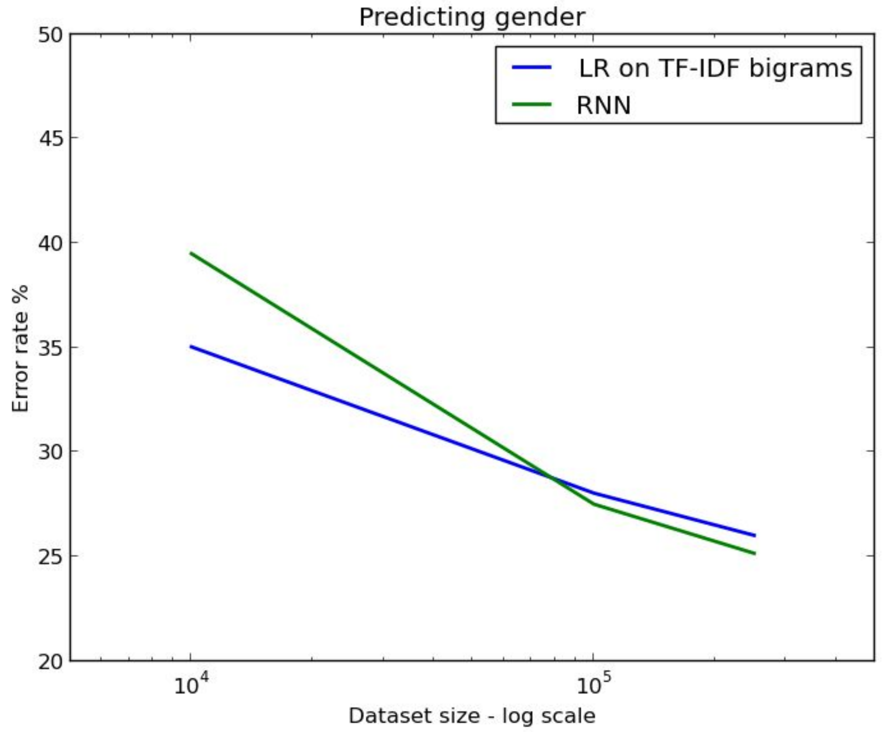
The library - Passage

- Tiny RNN library built on top of Theano
- https://github.com/IndicoDataSolutions/Passage
- Incredibly alpha =P
- We're working on it.
- Supports simple, LSTM, and GRU recurrent layers
- Supports multiple recurrent layers
- Supports deep input to and deep output from hidden layers
    - no deep transitions currently
- Supports embedding and onehot input representations
- Can be used for both regression and classification problems
    - Regression needs preprocessing for stability - working on it
- Much more in the pipeline

The dataset

- A scrape of blogger circa 2004.
- Originally from http://u.cs.biu.ac.il/~koppel/BlogCorpus.htm
- Just cleaned up into a csv at http://goo.gl/EbWA1u
- A little less than 300K blog post by 2K bloggers
- We’re going to use RNNs to predict gender given a blog post

Results



~75% accuracy, not too bad!

Haven't had nearly as much time to explore either model on this dataset...

Looks like similar trend compared to previous datasets! Linear models generalize much better for smaller amounts of data while RNNs begin to establish a lead as the dataset gets bigger.

Summary

- RNNs look to be a competitive tool in certain situations for text analysis.
- Especially if you have a large 1M+ example dataset
    - But a GPU is almost essential
- Otherwise, currently it can be difficult to justify compared to linear models
    - Speed
    - Complexity
    - Poor generalization with small datasets

Questions?

Contact

alec@indico.io