APS360 Fundamentals of AI

Lisa Zhang

Lecture 10; June 10, 2019

Logistics

- Lab 3 submit late by Wednesday
- Lab 4 due Sunday
- Midterm next Thursday
- We'll talk about the project this Thursday

Language Modelling

- Text Understanding
  - Question Answering
  - Sentiment Analysis
- Text Generation
  - Sentence completion
  - Generating captions, sentences, stories...
  - Translation
- ...and more!

Working with Text

Q: How is working with text different (more challenging) from working with images?

Working with Text

Q: How is working with text different (more challenging) from working with images?

- Grammar, spelling
- Many words to learn
- Choice of working with words vs characters (in English)
- Arbitrary length input / output

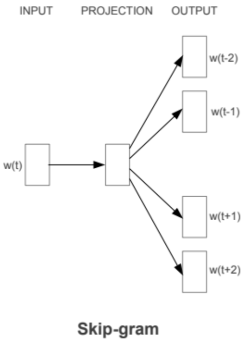
Agenda

- Review word2vec and GloVe embeddings
- Working with word embeddings
- Simple sentiment analysis model
- Recurrent neural networks

GloVe Embeddings

Training Word Embeddings

Key idea: the meaning of a word depends on its *context*, or other words that appear *nearby*



Q: True/False - words with similar word2vec/GloVe embeddings always have similar meanings.

Distance Measures

In order to talk about which words have "similar" GloVe embeddings, We need to introduce a measure of **distance** in the embedding space.

- Euclidean Distance
- Cosine Similarity

Euclidean Distance

The **Euclidean distance** of two vectors $x = [x_1, x_2, ...x_n]$ and $y = [y_1, y_2, ...y_n]$ is the 2-norm of their difference $x - y$:

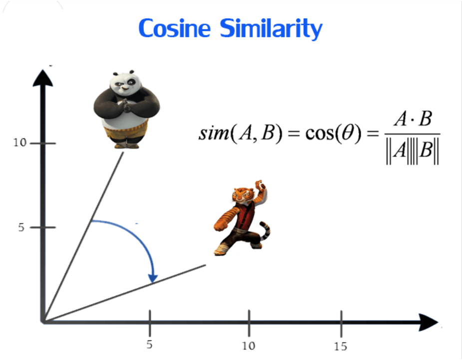
$$\sqrt{\sum_i (x_i - y_i)^2}$$

This is probably the distance measure you are most familiar with.

Q: What is the Euclidean distance between the vectors $x = [0, 1]$ and $y = [0, 2]$?

Cosine Similarity

The **cosine similarity** of two vectors x and y is the cosine of the angle between the two vectors.



Cosine similarity is useful when we want a distance measure that is **invariant to the magnitude of the vectors**.

Q: What is the cosine similarity between the vectors $x = [0, 1]$ and $y = [0, 2]$?

Computing Distances in PyTorch

Euclidean Distance:

```
torch.norm(glove['cat'] - glove['cat'])
```

Cosine Similarity:

```
torch.cosine_similarity(glove['cat'].unsqueeze(0), # need e
                        glove['dog'].unsqueeze(0))
```

Let's look at similarities between word embeddings in PyTorch.

Word analogies

One surprising thing about the embedding space is the extent of its **structure**.

We often see relationships like this in GloVe embeddings:

$$king - man + woman \approx queen$$

Bias in Word Embeddings

These word anaologies show that **machine learning models are not unbiased**

$$doctor - man + woman \approx ??$$

(See code)

Machine learning models learn the biases present in the data it is trained on.

Sentiment Analysis

Goal

Given a piece of text, identify the sentiment that the text conveys.

Can be applied to:

- movie reviews
- feedback
- emails
- tweets

garry connor @groovygarry · 3h

@Ryanair I have to say, I've just landed into Lisbon on flight FR1884, the whole experience was seamless from beginning to end! Everyone was extremely helpful but in particular your cabin crew who were friendly and professional! Very impressed!!!

1

Ryanair
@Ryanair

Replying to @groovygarry

Hi Garry, sorry for the inconveniences.
Please, submit a complaint here:
contactform.ryanair.com
Ani

Challenges

- Difficult problem in general
- Hard to collect clean, labelled data

Sentiment 140

- 160,000 tweets
- Sentiment determined by emoticon
- Collected by students doing a course project

Q: What are the advantages of using tweets as training data?

Q: What are the challenges of using tweets as training data?

Encoding the Input Data

For each tweet in the training data, we will

1. Split the tweet into words
2. Look up the GloVe embedding for each word, ignoring words that don't have embeddings
3. Add up the word embeddings to obtain an **embedding for the entire tweet**
4. The tweet embedding will be the input to a neural network

Splitting Into Words

> *I have to say, I've just landed into Lisbon on flight FR1884, the whole experience was seamless from the beginning to end! Everyone was extremely helpful but in particular your cabin crew who were friendly and professional! Very impressed!!!*

Words:

- i
- have
- to
- ...
- ...

Word Embeddings

I have to say, I've just landed into Lisbon on flight FR1884, the whole experience was seamless from the beginning to end! Everyone was extremely helpful but in particular your cabin crew who were friendly and professional! Very impressed!!!

Look up GloVe Embeddings

- i: = `tensor([ 1.1891e-01,  1.5255e-01, ...])`
- have: = `tensor([ 0.9491, -0.3497,  0.4812, ...])`
- to: = `tensor([ 0.6805, -0.0393,  0.3019, ...])`
- ...
- ...

Tweet Embedding

I have to say, I've just landed into Lisbon on flight FR1884, the whole experience was seamless from the beginning to end! Everyone was extremely helpful but in particular your cabin crew who were friendly and professional! Very impressed!!!

Add up embedding:

- i: = `[ 1.1891e-01,  1.5255e-01, ...]`
- have: = `[ 0.9491, -0.3497,  0.4812, ...]`
- to: = `[ 0.6805, -0.0393,  0.3019, ...]`
- ...
- **Tweet Embedding:** = `[ 16.183,  2.2113, ...]`

Training a Neural Network

- We will pre-compute tweet embeddings of all our training, validation, and test data (like Lab 3 transfer learning)
- Each tweet is represented by an embedding vector, which we put into a `DataLoader`
- Our neural network will be fully-connected

Neural Network Architecture

- Classifying “happy” vs “sad” is a binary classification problem
- However, we will use two output neurons and CrossEntropyLoss (instead of one output neuron and BCELoss)
- This architecture has a little more weights, and is usually easier to train (gets to better performance faster)

Considerations

Q: What are the advantages / disadvantages of this architecture?

Q: What are the advantages / disadvantages of using this dataset?

Q: What are ethical considerations that arise from building this model and using this dataset?

Q: What are ethical considerations that arise from using GloVe vectors?

Code!

Let's write some code!

Limitations

These two sentences will have the same embedding in our model:

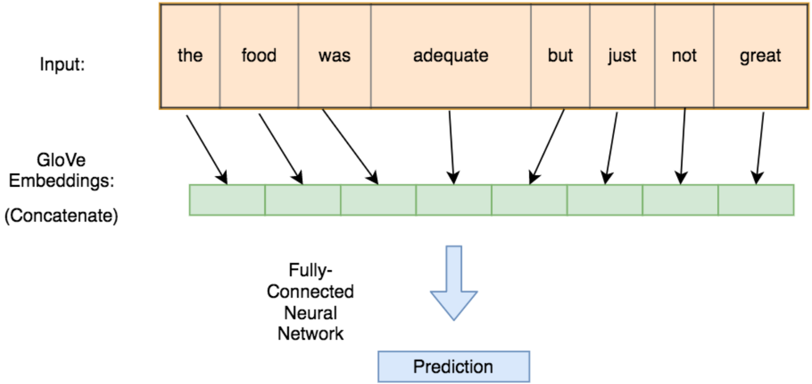
- The food was adequate, but just not great
- The food as not just adequate, but great

... but they have drastically different meanings.

Our model does not take into account the **order** of words

Idea #1

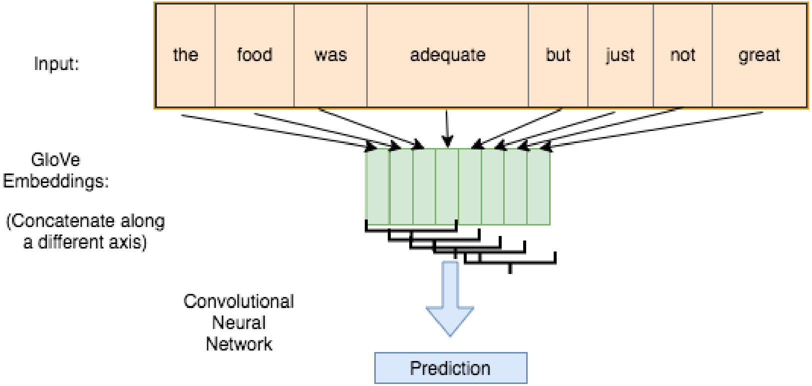
Concatenate (and flatten) the word embeddings, then train a neural network that takes the concatenated embedding as input.



Q: What is a drawback of this idea?

Idea #2

Concatenate the word embeddings, then train a convolutional neural network that takes the concatenated embedding as input.



Q: What is a drawback of this idea?

Recurrent Neural Networks

Want:

An architecture that

- Can take in variable-sized **sequential** input
- Can remember things over time: has some sort of **memory** or **state**

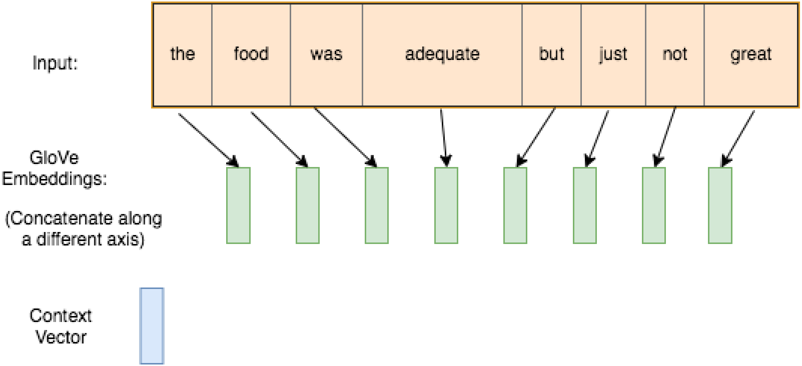
Want:

An architecture that

- Can take in variable-sized **sequential** input
- Can remember things over time: has some sort of **memory** or **state**

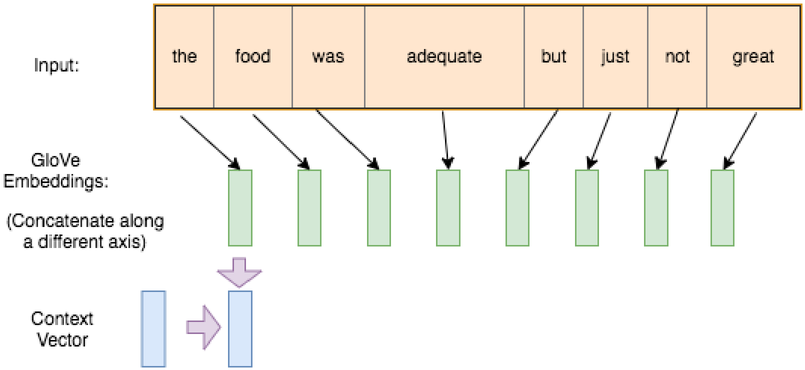
Recurrent Neural Networks!

RNN: Initial Hidden State



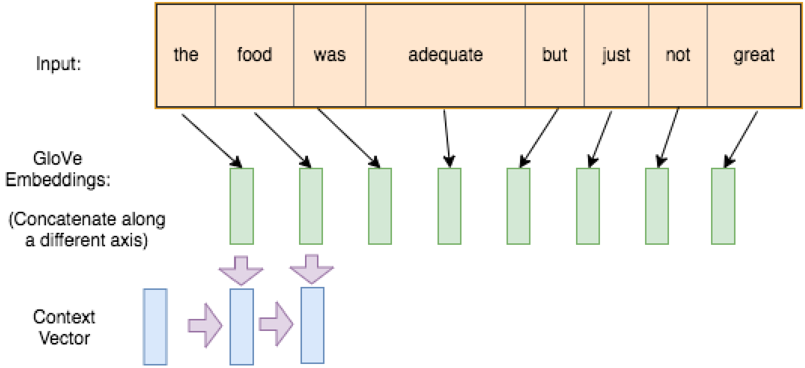
- Start with an initial **hidden state** with a blank slate (can be a vector of all zeros)

RNN: Update Hidden State



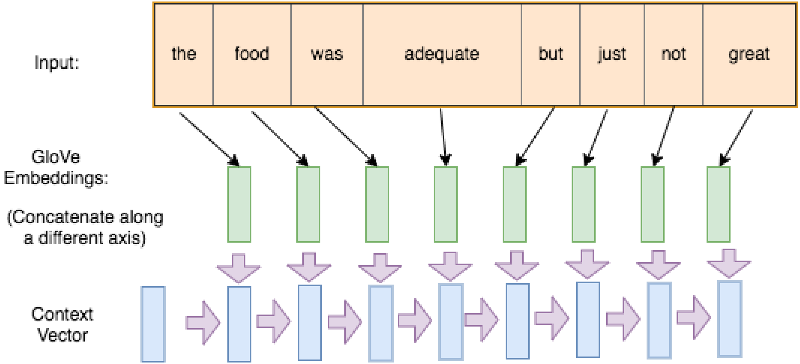
- Hidden state is updated based on previous hidden state, and the input
- `hidden = update_function(hidden, input)`

RNN: Continue Updating Hidden State



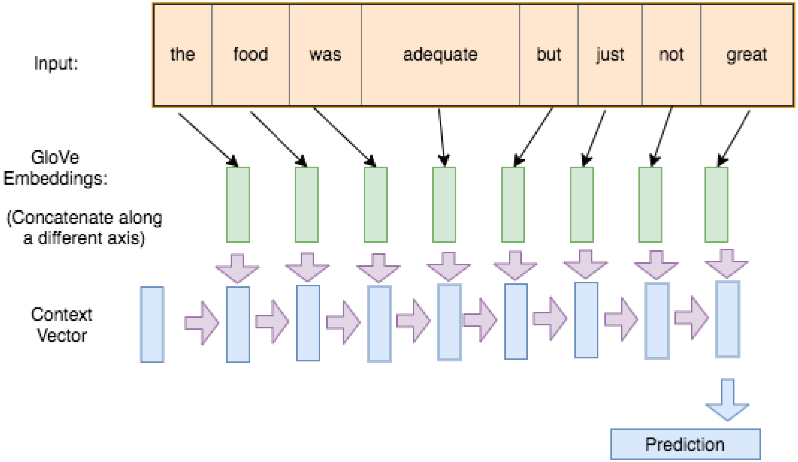
- Hidden state is updated based on previous hidden state, and the input **using the same neural network as before** (weight sharing).
- `hidden = update_function(hidden, input)`

RNN: Last Hidden State



- Continue updating the hidden state until we run out of tokens.
- `hidden = update_function(hidden, input)`

RNN: Compute Prediction



- Use the last hidden state as input to a prediction network
- `output = prediction_function(hidden)`
- Alternatively, max-pool or average-pool over all computed hidden states.

RNN Notes

- Not all Recurrent Neural Networks use GloVe embeddings to represent the input tokens
- We could have used a one-hot encoding
  - Not practical with this example
  - Possible with a small **vocabulary size** (total # uniq tokens)
- As an example, we will work with **Character-level RNNs** in lab 5

RNN Expectations

- You are not expected to know about the computations that happen when a hidden state is updated
- You are expected to know how to use a Recurrent Neural Network in PyTorch
- Let's take a look!

Batching

- When trainign an RNN, sequences in each batch must all have the same length
- So, sequences that are shorter needs to be padded
- In practice, try to batch similar-length training examples to minimize padding

Learning Long-Term Dependencies

- Historically, Recurrent Neural Networks were hard to train
- Better RNN units for learning long-term dependencies:
  - Long Short-Term Memory (LSTM): requires an extra **cell state**
  - Gated Recurrent Unit (GRU): only requires a hidden state

GloVe Embedding

Q: What would need to change if we want to use an 100-dimensional GloVe embedding?

```
class TweetRNN(nn.Module):
 def __init__(self, input_size, hidden_size, num_classes):
  super(TweetRNN, self).__init__()
  self.emb = nn.Embedding.from_pretrained(glove.vectors)
  self.rnn = nn.RNN(input_size,
                    hidden_size,
           batch_first=True)
  self.fc = nn.Linear(hidden_size, num_classes)

 def forward(self, x):
  x = self.emb(x)
  out, _ = self.rnn(x)
  out = self.fc(out[:, -1, :])
  return out
```