Lecture 10:
Machine Translation,
Sequence-to-sequence and Attention

Abigail See
Announcements

• Honor code issues: Assignment 2

• Assignment 3 released

• Azure credits released

• Default final project update:
  • New handout released
  • Submission instructions released

• Custom final project: you should receive feedback on your proposal this week

• Midterm grades: released after lecture
Happy Valentines Day!

RNN-generated candy hearts

Welcome to the second half of the course!

• Remaining lectures are mostly geared towards projects

• We’ll bring you to the cutting-edge of NLP+DL research

• Lectures will be more high-level
  • No more gradient computations!
  • Sometimes we’ll sketch an overview – if you’re interested in a topic, you can read more after class

• However: today’s lecture will cover two core NLP Deep Learning techniques
Overview

Today we will:

• Introduce a new task: Machine Translation

• Introduce a new neural architecture: sequence-to-sequence

• Introduce a new neural technique: attention
**Machine Translation**

**Machine Translation (MT)** is the task of translating a sentence $x$ from one language (the **source language**) to a sentence $y$ in another language (the **target language**).

$x$: \( L'homme \text{ est n}é \text{ libre, et partout il est dans les fers} \)

\[\downarrow\]

$y$: \( Man \text{ is born free, but everywhere he is in chains} \)
1950s: Early Machine Translation

Machine Translation research began in the early 1950s.

• Mostly Russian → English (motivated by the Cold War!)

• Systems were mostly rule-based, using a bilingual dictionary to map Russian words to their English counterparts
  • A cool by-product: Quicksort!

Source: https://youtu.be/K-HfpsHPmvw
1990s-2010s: Statistical Machine Translation

- **Core idea:** Learn a probabilistic model from data
- Suppose we’re translating French → English.
- We want to find best English sentence \( y \), given French sentence \( x \)

\[
\text{argmax}_y P(y|x)
\]

- Use Bayes Rule to break this down into two components to be learnt separately:

\[
= \text{argmax}_y P(x|y)P(y)
\]

**Translation Model**
Models how words and phrases should be translated.
Learnt from parallel data.

**Language Model**
Models how to write good English.
Learnt from monolingual data.
1990s-2010s: Statistical Machine Translation

- **Question**: How to learn translation model $P(x|y)$?
- First, need large amount of **parallel data** (e.g. pairs of human-translated French/English sentences)

![The Rosetta Stone]

- Ancient Egyptian
- Demotic
- Ancient Greek
1990s-2010s: Statistical Machine Translation

- **Question**: How to learn translation model $P(x|y)$?
- First, need large amount of **parallel data** (e.g. pairs of human-translated French/English sentences)

- Break it down further: we actually want to consider

$$P(x, a|y)$$

where $a$ is the **alignment**, i.e. word-level correspondence between French sentence $x$ and English sentence $y$
What is alignment?

Alignment is the correspondence between particular words in the translated sentence pair.

- Note: Some words have no counterpart

![Alignment Diagram]

- Japan
- shaken
- by
- two
- new
- quakes
- Japon
- secoué
- par
- deux
- nouveaux
- séismes

**“spurious” word**

Le

- Le
- Japon
- secoué
- par
- deux
- nouveaux
- séismes

Japan shaken by two new quakes

Japan shaken by two new quakes

Le programme a été mis en application

Le Japon secoué par deux nouveaux séismes
Alignment is complex

Alignment can be one-to-many (these are “fertile” words)

And the program has been implemented

Le programme a été mis en application

Statistical MT

Pioneered at IBM in the early 1990s

Let’s make a probabilistic model of translation

\[ P(e | f) \]

Suppose \( f \) is

- \( \text{de rien} \)
- \( \text{you’re welcome} \)
- \( \text{nothing} \)
- \( \text{piddling} \)
- \( \text{underpants} \)

\[ P(\text{you’re welcome} | \text{de rien}) = 0.45 \]
\[ P(\text{nothing} | \text{de rien}) = 0.13 \]
\[ P(\text{piddling} | \text{de rien}) = 0.01 \]
\[ P(\text{underpants} | \text{de rien}) = 0.000000001 \]

Hieroglyphs

Statistical Solution

- Parallel Texts
  - Rosetta Stone
  - Demotic
  - Greek
- Statistical Solution
  - Instruction Manuals
  - Hong Kong/Macao Legislation
  - Canadian Parliament Hansards
  - United Nations Reports
  - Official Journal of the European Communities
  - Translated news

Hmm, every time one sees “banco”, translation is “bank” or “bench” … If it’s “banco de…”, it always becomes “bank”, never “bench”…

A Division of Labor

Spanish
Broken English
English

Spanish/English Bilingual Text

Statistical Analysis

Que hambre tengo yo

I am so hungry

Translation Model

\[ P(f | e) \]

Language Model

\[ P(e) \]

Decoding algorithm

\[ \text{argmax} \ P(f | e) \times P(e) \]

What hunger have I,

Hungry I am so,

I am so hungry,

Have I that hunger …

Fidelity

Fluency

Alignments

We can factor the translation model by identifying alignments (correspondences) between words in \( f \) and words in \( e \)

- \( \text{Japan shaken by two new quakes} \)

- \( \text{Le Japon secoué par deux nouveaux séismes} \)

- \( \text{spurious word} \)

Alignments: harder

And the program has been implemented

Le programme a été mis en application

zero fertility word

not translated

one-to-many alignment
Alignment is complex

Alignment can be many-to-one

The balance was the territory of the aboriginal people

Le reste appartait aux autochtones

The balance was the territory of the aboriginal people

Les pauvres sont démunis

Many-to-one alignments

Alignment as a vector

Mary did not slap the green witch

Maria no aba una botefada a la bruja verde

Choose length J for French sentence

For each j in 1 to J:

- Choose a_j uniformly from 0, 1, ..., I
- Choose f_j by translating e_a_j

Given English sentence e_1, e_2, ..., e_I:

We want to learn how to do this

Want: P(f|e)

IBM Model 1 parameters

And the program has been implemented

Le programme a été mis en application

Applying Model 1*

As translation model

As alignment model

P(f, a|e) can be used as a translation model or an alignment model

* Actually, any P(f, a|e), e.g., any IBM model
Alignment is complex

Alignment can be many-to-many (phrase-level)

The poor don't have any money

Les pauvres sont démunis

Alignment as a vector

Mary did not slap the green witch

Le reste appartenait aux aborigènes

Alignments: hardest

The poor don’t have any money

Les pauvres sont démunis

Phrase alignment

For each j in 1 to J:

- Choose \( a_j \) uniformly from \( 0, 1, \ldots, I \)
- Choose \( f_j \) by translating \( e_{a_j} \)

We want to learn how to do this

Want: \( P( f \mid e ) \)

Mathematics used in all IBM models

\( a_j = 0 \) if \( f_j \) is spurious

No one-to-many alignments

No many-to-many alignments

But provides foundation for phrase-based alignment

IBM Model 1 parameters

And the program has been implemented

Le programme a été mis en application

Applying Model 1*

As translation model

As alignment model

\( P( f, a \mid e ) \) can be used as a translation model or an alignment model

* Actually, any \( P( f, a \mid e ) \), e.g., any IBM model
1990s-2010s: Statistical Machine Translation

- **Question**: How to learn translation model $P(x|y)$?
- First, need large amount of parallel data (e.g. pairs of human-translated French/English sentences)
- Break it down further: we actually want to consider
  
  $$P(x, a|y)$$

  where $a$ is the alignment, i.e. word-level correspondence between French sentence $x$ and English sentence $y$

- We learn $P(x, a|y)$ as a combination of many factors, including:
  - Probability of particular words aligning
    - Also depends on position in sentence
  - Probability of particular words having particular fertility
  - Etc.
1990s-2010s: Statistical Machine Translation

\[ \arg \max_y P(x|y) P(y) \]

Question: How to compute this argmax?

• We could enumerate every possible \( y \) and calculate the probability? → Too expensive!

• **Answer:** Use a heuristic search algorithm to gradually build up the the translation, discarding hypotheses that are too low-probability
Searching for the best translation

• Task: translate this sentence from German into English
  er geht ja nicht nach hause
  er geht ja nicht
  he does not go
  nach hause
Searching for the best translation

<table>
<thead>
<tr>
<th>er</th>
<th>geht</th>
<th>ja</th>
<th>nicht</th>
<th>nach</th>
<th>hause</th>
</tr>
</thead>
<tbody>
<tr>
<td>he</td>
<td>is</td>
<td>yes</td>
<td>not</td>
<td>after</td>
<td>house</td>
</tr>
<tr>
<td>it</td>
<td>are</td>
<td>is</td>
<td>do not</td>
<td>to</td>
<td>home</td>
</tr>
<tr>
<td>, it</td>
<td>goes</td>
<td>, of course</td>
<td>does not</td>
<td>according to</td>
<td>chamber</td>
</tr>
<tr>
<td>, he</td>
<td>go</td>
<td>,</td>
<td>is not</td>
<td>in</td>
<td>at home</td>
</tr>
</tbody>
</table>

Many translation options to choose from

– in Europarl phrase table: 2727 matching phrase pairs for this sentence

– by pruning to the top 20 per phrase, 202 translation options remain

Chapter 6: Decoding

Decoding: Find Best Path

backtrack from highest scoring complete hypothesis
1990s-2010s: Statistical Machine Translation

- SMT is a huge research field
- The best systems are extremely complex
  - Hundreds of important details we haven’t mentioned here
  - Systems have many separately-designed subcomponents
- Lots of feature engineering
  - Need to design features to capture particular language phenomena
- Require compiling and maintaining extra resources
  - Like tables of equivalent phrases
- Lots of human effort to maintain
  - Repeated effort for each language pair!
2014

MT research (dramatic reenactment)

Neural Machine Translation
What is Neural Machine Translation?

• Neural Machine Translation (NMT) is a way to do Machine Translation with a *single neural network*

• The neural network architecture is called *sequence-to-sequence* (aka *seq2seq*) and it involves *two RNNs*. 
Neural Machine Translation (NMT)

The sequence-to-sequence model

Encoding of the source sentence. Provides initial hidden state for Decoder RNN.

Encoder RNN produces an encoding of the source sentence.

Decoder RNN is a Language Model that generates target sentence conditioned on encoding.

Note: This diagram shows test time behavior: decoder output is fed in as next step's input.
Neural Machine Translation (NMT)

• The sequence-to-sequence model is an example of a Conditional Language Model.
  • **Language Model** because the decoder is predicting the next word of the target sentence $y$
  • **Conditional** because its predictions are *also* conditioned on the source sentence $x$

• NMT directly calculates $P(y|x)$:

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

  Probability of next target word, given target words so far and source sentence $x$

• **Question**: How to train a NMT system?
• **Answer**: Get a big parallel corpus...
Training a Neural Machine Translation system

\[
J = \frac{1}{T} \sum_{t=1}^{T} J_t = J_1 + J_2 + J_3 + J_4 + J_5 + J_6 + J_7
\]

Encoder RNN

\[
\text{les pauvres sont démunis}
\]

Source sentence (from corpus)

Decoder RNN

\[
<\text{START}> \text{ the poor don’t have any money}
\]

Target sentence (from corpus)

Seq2seq is optimized as a single system. Backpropagation operates “end to end”.

Backpropagation operates “end to end”.

Note: The diagram and equation represent a training objective for a seq2seq model, where each term in the loss function corresponds to a different aspect of the translation process, such as the probability of specific words or sequences.
Better-than-greedy decoding?

- We showed 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?**
Better-than-greedy decoding?

- Greedy decoding has no way to undo decisions!
  - *les pauvres sont démunis (the poor don’t have any money)*
  - → *the ____*
  - → *the poor ____*
  - → *the poor are ____*

- Better option: use **beam search** (a search algorithm) to explore several hypotheses and select the best one
Beam search decoding

• Ideally we want to find $y$ that maximizes
  
  $$P(y|x) = P(y_1|x) P(y_2|y_1, x) P(y_3|y_1, y_2, x) \ldots, P(y_T|y_1, \ldots, y_{T-1}, x)$$

• We could try enumerating all $y \rightarrow$ too expensive!
  • Complexity $O(V^T)$ where $V$ is vocab size and $T$ is target sequence length

• **Beam search**: On each step of decoder, keep track of the $k$ most probable partial translations
  • $k$ is the beam size (in practice around 5 to 10)
  • Not guaranteed to find optimal solution
  • But much more efficient!
Beam search decoding: example

Beam size = 2
Beam search decoding: example

Beam size = 2
Beam search decoding: example

Beam size = 2

<START> → a

the → poor

poor → people

poor → person
Beam search decoding: example

Beam size = 2
Beam search decoding: example

Beam size = 2

the

poor

people

<START>

a

poor

person

are
don’t

not

always

have

take

person

but
Beam search decoding: example

Beam size = 2

```
<START> a poor person
the poor people
always not are don’t have take
in with any enough
```
Beam search decoding: example

Beam size = 2

<START> a poor person poor person

the poor people

are don’t

always not

have take

in with

any enough

money funds

money funds
Beam search decoding: example

Beam size = 2

<START> → the → poor people → don’t → are → not → always

→ in

→ with

→ have

→ take

→ any

→ enough

→ money

→ funds

→ money

→ funds

→ a

→ poor person

→ but

→ person
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!
How do we evaluate Machine Translation?

**BLEU (Bilingual Evaluation Understudy)**

- BLEU compares the *machine-written translation* to one or several *human-written translation(s)*, and computes a *similarity score* based on:
  - *n*-gram precision (usually up to 3 or 4-grams)
  - 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 😞
MT progress over time
[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal]

NMT: the biggest success story of NLP Deep Learning

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

- **2014**: First seq2seq paper published
- **2016**: Google Translate switches from SMT to NMT
- **This is amazing!**
  - **SMT** systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by a handful of engineers in a few months
So is Machine Translation solved?

• **Nope!**

• Many difficulties remain:
  • Out-of-vocabulary words
  • Domain mismatch between train and test data
  • Maintaining context over longer text
  • Low-resource language pairs
So is Machine Translation solved?

- **Nope!**
- **Using common sense** is still hard
So is Machine Translation solved?

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

Source: https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c
So is Machine Translation solved?

- Nope!
- **Uninterpretable systems do strange things**

Source: http://languagelog.ldc.upenn.edu/nll/?p=35120#more-35120
NMT research continues

NMT is the **flagship task** for NLP Deep Learning

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

- In **2018**: NMT research continues to **thrive**
  - Researchers have found **many, many improvements** to the “vanilla” seq2seq NMT system we’ve presented today
  - But **one improvement** is so integral that it is the new vanilla...

**ATTENTION**
Sequence-to-sequence: the bottleneck problem

Encoding of the source sentence.

Source sentence (input)

Target sentence (output)

Problems with this architecture?
**Sequence-to-sequence: the bottleneck problem**

Encoding of the source sentence. This needs to capture *all information* about the source sentence. Information bottleneck!

**Source sentence (input)**

- *les pauvres sont démunis*

**Target sentence (output)**

- *the poor don’t have any money*
Attention

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

- **Core idea**: on each step of the decoder, *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

Source sentence (input)

Encoder RNN

Attention scores

dot product

Decoder RNN

les pauvres sont démunis

<START>
Sequence-to-sequence with attention

Encoder RNN

Attention scores

Decoder RNN

dot product

Source sentence (input)
Sequence-to-sequence with attention
Sequence-to-sequence with attention

Source sentence (input)

les pauvres sont démunis

<START>

dot product
Sequence-to-sequence with attention

On this decoder timestep, we’re mostly focusing on the first encoder hidden state (”les”)

Take softmax to turn the scores into a probability distribution
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 the hidden states that received high attention.
Sequence-to-sequence with attention

Encoder RNN

Attention scores

Attention distribution

Attention output

Decoder RNN

the

\( \hat{y}_1 \)

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

Source sentence (input)

les pauvres sont démunis

<START>
Sequence-to-sequence with attention

- **Encoder RNN**: les pauvres sont démunis
- **Attention scores**: Attention distribution
- **Attention output**: 
- **Decoder RNN**: 
- **Attention scores**: 
- **Attention output**: poor
- **Output**: \( \hat{y}_2 \)
Sequence-to-sequence with attention

Source sentence (input):

- les
- pauvres
- sont
- démunis

<START> the poor

Encoder RNN

Attention scores

Attention distribution

Attention output

Decoder RNN

Attention scores

Attention distribution

Attention output

\( \hat{y}_2 \)

\( don't \)
Sequence-to-sequence with attention

Encoder RNN

Source sentence (input)

les pauvres sont démunis

<START> the poor don’t

Decoder RNN

Attention scores

Attention distribution

Attention output

have

\( \hat{y}_2 \)
Sequence-to-sequence with attention

Encoder RNN

Attention scores

Attention distribution

Attention output

any

Decoder RNN

any

les pauvres sont démunis

<START> the poor don't have

Source sentence (input)
Sequence-to-sequence with attention

Source sentence (input):

les pauvres sont démunis

Encoder RNN

Attention distribution

Attention scores

<START> the poor don’t have any

Decoder RNN

Attention output

Attention scores

Attention distribution

money

\( \hat{y}_2 \)
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 $e^t$ for this step:
  $$e^t = [s^T_t h_1, \ldots, s^T_t h_N] \in \mathbb{R}^N$$
- We take softmax to get the attention distribution $\alpha^t$ for this step (this is a probability distribution and sums to 1)
  $$\alpha^t = \text{softmax}(e^t) \in \mathbb{R}^N$$
- We use $\alpha^t$ to take a weighted sum of the encoder hidden states to get the attention output $a_t$
  $$a_t = \sum_{i=1}^{N} \alpha^t_i h_i \in \mathbb{R}^h$$
- Finally we concatenate the attention output $a_t$ with the decoder hidden state $s_t$ and proceed as in the non-attention seq2seq model
  $$[a_t; s_t] \in \mathbb{R}^{2h}$$
Attention is great

- Attention significantly improves NMT performance
  - It’s very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself

Alignments: harder
- The balance was the territory of the aboriginal people
- The poor don’t have any money

Alignments: hardest
- The poor don’t have any money

Phrase alignment
- IBM Model 1 generative story

Choose length $J$ for French sentence
- For each $j$ in 1 to $J$:
  - Choose $a_j$ uniformly from 0, 1, …, $I$
  - Choose $f_j$ by translating $e_{aj}$

We want to learn how to do this
- Want: $P(f|e)$

IBM Model 1 parameters
- And the program has been implemented
- We get alignment for free!
- This is cool because we never explicitly trained an alignment system
- The network just learned alignment by itself

Applying Model 1*
- As translation model
- As alignment model

$P(f, a|e)$ can be used as a translation model or an alignment model

* Actually, any $P(f, a|e)$, e.g., any IBM model
Recap

• We learned the history of Machine Translation (MT)

• Since 2014, Neural MT rapidly replaced intricate Statistical MT

• Sequence-to-sequence is the architecture for NMT (uses 2 RNNs)

• Attention is a way to focus on particular parts of the input
  • Improves sequence-to-sequence a lot!
Sequence-to-sequence is versatile!

- Sequence-to-sequence is useful for more than just MT

- Many NLP tasks can be phrased as sequence-to-sequence:
  - Summarization (long text → short text)
  - Dialogue (previous utterances → next utterance)
  - Parsing (input text → output parse as sequence)
  - Code generation (natural language → Python code)
Next time

• More *types* of attention

• More *uses* for attention

• More advanced sequence-to-sequence techniques