EXERCISES 25

DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

25.1 Word Embeddings

Exercise 25.WEVZ

Run a word embedding visualization tool such as `http://projector.tensorflow.org/` and try to get a feel for how the embeddings work: what words are near each other? Are there surprises for unrelated words that are near or related words that are far apart? Report on your findings. Consider:

a. Common concrete nouns like "people" or "year" or "dog."

b. The most common words, which carry little meaning: "the," "be," "of," "to," etc.

c. Abstract concepts like "configuration" or "topological."

d. Words that are part of sequential series such as numbers, months, presidents, days of the week (is "Monday" closer to "Sunday" or "Tuesday" or "Friday"?).

e. Words that are part of unordered groups such as "France" or "Maine" or "lion."

f. Ambiguous words (is "Turkey" grouped with countries or birds?).

g. UMAP versus T-SNE versus PCA visualizations.

h. What words are at the periphery of the embedding? Near the center? Is that significant? Try changing the principal components that are mapped to each of the XYZ axes in PCA and see what difference that makes.

i. What else can you explore?

Each student will take their own path in exploring this question.

Exercise 25.WEMD

Run a notebook to generate a word embedding model such as `https://www.tensorflow.org/text/guide/word_embeddings`, which trains an embedding model based on a corpus of IMDB movie reviews. Create the embedding model and visualize the embeddings. Then create another model based on a different text corpus and compare. What similarities and differences do you notice?

If you are not able to run a notebook, you can compare five different pre-

built word embeddings at `http://vectors.nlpl.eu/explore/embeddings/en/associates/`.

The specific vocabulary of movies, especially the names of actors and directors, will not be present in another corpus. However for two large corpora, many of the most common words will look similar in the embedding model.

Exercise 25.WEAB

Examine how well word embedding models can answer analogy questions of the form "**A** is to **B** as **C** is to [what]?" (e.g. "Athens is to Greece as Oslo is to Norway") using vector arithmetic. Create your own embeddings or use an online site such as `https://lamyiowce.github.io/word2viz/` or `http://bionlp-www.utu.fi/wv_demo/`.

a. What analogies work well, and what ones fail?

b. In handling Country:Capital analogies, is there a problem with ambiguous country names, like "Turkey"?

c. Do analogies start to fail for rarer words? For example, "one is to 1 as two is to [what]?" will reliably retrieve "2," but how well that hold for "ninety" or "thousand"?

d. What research papers can help you understand word embedding analogies?

e. What else can you explore?

Each student will take their own path in exploring this question. In general, it seems like analogies work best when there is a word affix (such "great is to greatest as tall is to tallest") or when the words have very specific references in the real world (like "Oslo") that are likely to have been mentioned in the training corpus, or when words naturally occur in an order (such as "one, two, three"). However, counting or arithmetic analogies only hold for the first few integers. Word vectors can't represent everything there is to know about a word. Good papers include `https://aclanthology.org/S17-1001/` and `https://arxiv.org/abs/1901.09813`.

25.2 Recurrent Neural Networks for NLP

Exercise 25.RNCH

So far we've concentrated on word embeddings to represent the fundamental units of text. But it is also possible to have a **character-level model**. Read Andrej Karpathy's 2015 article *The Unreasonable Effectiveness of Recurrent Neural Networks* and download his `char-rnn` code (the PyTorch version at `https://github.com/jcjohnson/torch-rnn` is easier to use than the original version at `https://github.com/karpathy/char-rnn`). Train the model on text of your choice.

a. Show how the randomly generated text improves with more iterations of training.

b. Find any other interesting properties of the model on your text.
c. Replicate some of Karpathy's findings, such as the fact that some cells learn to count nesting level of parentheses.
d. Compare the character-level RNN model to a character-level n-gram model as described in Yoav Goldberg's *The unreasonable effectiveness of Character-level Language Models (and why RNNs are still cool)* at `https://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139`.
e. What do you see as the differences between RNN and n-gram models?

Each student will take their own path in exploring this question. The main difference between models is that the n-gram model by definition has no memory of context that precedes the previous n characters, while the RNN can learn to selectively remember parts of the context (such as indentation and nesting).

Exercise 25.RNTF

Which of the following are true or false?

a. A RNN is designed specifically for processing word sequences.
b. A RNN is designed to process any time-series data.
c. An RNN has a limit on how far into the past it can remember.
d. An n-gram model has a limit on how far into the past it can remember.
e. An RNN can look into the future.
f. An RNN must use LSTM units in order to copy information from one time step to the next.

a. False; RNN's work on character sequences as well as word sequences, and on any kind of time-series data.
b. True.
c. False. Information about any input can be passed on from one hidden layer to the next for any number of time steps.
d. True. An n-gram model can only look back n steps.
e. False. A single RNN cannot look at future inputs until their time arrives. But a bidirectional RNN, which consists of two RNNs put together, can work in both directions.
f. False. A regular RNN can copy information forward. The advantage of an LSTM is that it can learn what is important to copy, and what is not.

Exercise 25.RNPS

Apply an RNN to the task of **part of speech tagging**. Run a notebook (such as `https://www.kaggle.com/tanyadayanand/pos-tagging-using-rnn` or

https://github.com/roemmele/keras-rnn-notebooks/tree/master/pos_tagging), train the model on some tagged sentences, and evaluate it on some test data. Compare the results to a non-recurrent POS tagger, such as http://librarycarpentry.org/lc-tdm/13-part-of-speech-tagging-text/index.html.)

Students should achieve roughly 99% accuracy.

Exercise 25.RNTC

Apply an RNN to the task of **text classification**, in particular **binary sentiment analysis**: classifying movie reviews on IMDB as either positive or negative. Run a notebook such as https://www.tensorflow.org/text/tutorials/text_classification_rnn and report on your results.

a. What level of accuracy can you achieve?

b. Gather some reviews from another site, not IMDB, and see if the trained model performs as well on them.

c. How does accuracy vary with the length of the review?

d. Which words are most associated with a positive review? A negative review?

Each student will have their own approach to this exercise. To determine the most predictive (positive/negative) words, students can either examine the weights in their model, or they can vary the input: replace a word in the input with [UNK] and see how the predicted score changes. Repeat over all words.

25.3 Sequence-to-Sequence Models

Exercise 25.YAMS

This exercise is about the difficulty of translation, but does not use a large corpus, nor any complex algorithms–just your own ingenuity. A rare language spoken by only about a hundred people has an unusal number system that is not base ten. Below are the translations of the first ten cube numbers (1^3 to 10^3) in this language, in some random order. Can you identify which phrase is which number, and how the number system works?

a. eser tarumpao yuow ptae eser traowo eser

b. eser traowo yuow

c. naempr

d. naempr ptae eser traowo eser

e. naempr tarumpao yuow ptae yuow traowo naempr

f. naempr traowo yempoka

g. tarumpao
h. yempoka tarumpao yempoka ptae naempr traowo yempoka
i. yuow ptae yempoka traowo tampui
j. yuow tarumpao yempoka ptae naempr traowo yuow

The number phrases are from the Ngkolmpu language in New Guinea, which uses base six. Base six makes sense because their staple diet is the yam, and the yam's elongated shape means that six of them fit together neatly in a circle. The numbers in order are:

a. 1000, 27, 1, 64, 343, 8, 216, 512, 125, 729.

The vocabulary is:

a. $1 =$ naempr
b. $2 =$ yempoka
c. $3 =$ yuow
d. $4 =$ eser
e. $5 =$ tampui
f. $6 =$ traowo
g. $6^2 = 36 =$ ptae
h. $6^3 = 216 =$ tarumpao

This puzzle is due to Alex Bellos from his Monday math puzzle series at *The Guardian*.

a.

Exercise 25.SSMT

Experiment with online sequence-to-sequence neural machine translation model, such as `https://translate.google.com/` or `https://www.bing.com/translator` or `https://www.deepl.com/translator`. If you know two languages well, look at translations between them. If you don't translate into a language and then back to the language you do know.

a. How well does the system handle very rare words?
b. Does the system properly rearrange the order of words when necessary?
c. Do the translations encode biases?
d. How does the system do on very long sentences (over 60 words)?

The main point of this exercise is to gain experience with MT systems, not specifically to answer the four parts, but here are answers:

a. Rare words can be a problem. Systems that are character-based do better than strictly word-based systems at encoding things like verb endings.

b. The attention mechanism does consistently well on local word reordering (e.g. translating "red pencil" as "crayon rouge"), but can have difficulties, say, when the subject of a 30-word sentence appears first in the source language and last in the target language.

c. Some systems will encode the biases of their training corpora, for example assuming that the neutral phrase "the doctor" is translated to the masculine "el doctor" in Spanish. Other systems avoid this bias by offering two choices: "la doctora" and "el doctor."

d. Neural machine translation systems historically perform poorly on very long sentences. But perhaps there have been improvements by the time you do this exercise.

Exercise 25.SSOP

Besides machine translation, describe some other tasks that can be solved by sequence-to-sequence models.

Here are some possibilities:

a. Summarization: compressing a sentence, paragraph, or document into a shorter summary.

b. Conversation: Chatbots such as Google's Meena are built with seq2seq models.

c. Interpretation: Mathematical problems such as integrals and differential equations can be mapped to their solutions with seq2seq models.

d. Part of speech tagging: Seq2seq models may represent parts of speech latently as they go about their tasks, but they can be trained to output the POS labels.

e. Parsing: As with parts of speech, seq2seq models may be trained to output a dependency structure (e.g. the current word has a dependency link to the word 4 words ago) that represents a parse tree.

f. Morphology: A seq2seq model can be trained to invent the morphological form of a word with a specific purpose. For example, given the sequence "employ, NOUN, AGENT" it can output "employer," and given "employ, ADJ" it can output "employable."

g. Drug discovery: descriptions of molecules can be mapped to a feature vector in multidimensional space; the space of molecules can then be examined to see which are nearby and which have certain relations that might indicate they will be useful as drugs for some purpose.

25.4 The Transformer Architecture

Exercise 25.PRCV

Since the publication of the textbook, a new architecture called **Perceiver** was introduced by Jaegle *et al.* in their article *Perceiver: General Perception with Iterative Attention* `https://arxiv.org/abs/2103.03206`. Read the article and say which of the following are true or false.

a. Perceiver is a kind of RNN model.

b. Perceiver is a kind of CNN model.
c. Perceiver is a kind of seq2seq model.
d. Perceiver is a kind of Transformer model.
e. Perceiver was specifically designed for large data sets.
f. Perceiver was specifically designed for audio/video inputs.
g. Perceiver performs worse on computer vision tasks than a CNN model specifically designed for images, but Perceiver is more general.
h. Perceiver avoids overfitting.

a. Either True or False is an acceptable answer. A Perceiver can be interpreted as an RNN, with each layer in the network serving as a step. But it can also be seen as a version of a Transformer, which processes the input all at once, with positional encodings.
b. False. Perceiver does not use convolutional units; instead it directly attends to pixels.
c. True. Perceiver is a seq2seq model.
d. True. Perceiver is a variant of the Transformer model.
e. True. Perceiver has minimal inductive biases and wants the data to speak for itself; it is designed to be efficient with billions of examples, each with 100,000 input features.
f. False. Perceiver handles those modalities, but the architecture does not bias towards any particular type of input.
g. False. Perceiver's performance on image tasks is comparable to a CNN model, not worse.
h. False. Perceiver does indeed perform well on a variety of tasks, but its lack of inductive bias means it is susceptible to overfitting.

Exercise 25.TRMT

Run a tutorial notebook such as `https://www.tensorflow.org/text/tutorials/transformer` to train a Transformer model on a bilingual corpus to do machine translation. Test it to see how it performs. How does it handle unknown words?

The given tutorial does Portuguese/English translation, but other language pairs are possible. Following through the notebook should introduce students to all parts of the Transformer architecture. The model should mostly transliterate unknown words, but sometimes will change the spelling of an unknown word to match the letter frequencies of the target language, and will sometimes latch on to parts of more familiar words, as in translating "triceratops" to "trijacopters."

Exercise 25.TRSW

We have considered recurrent models that work a word at a time, and models that work a character at a time. It is also possible to use a **subword representation**, in which, say, the

word "searchability" is represented as two tokens, the word "search" followed by the suffix "-ability."

Run a notebook such as `https://www.tensorflow.org/text/guide/subwords_tokenizer` that allows you to build a subword tokenizer. Train it on some text. What are the tokens that get represented? What words and what nonwords? How can you visualize the results?

It should be straightforward to run the given notebook. Common words get their own representation. Uncommon words or long words with common prefixes or suffixes are broken into subwords. Punctuation gets its own tokens. Helpful visualizations include plots that consider whether a token is a word or non-word, how frequently it occurs, and how many letters it contains.

25.5 Pretraining and Transfer Learning

Exercise 25.TFTR

Run a notebook such as `https://www.tensorflow.org/hub/tutorials/tf2_text_classification` that loads a pre-trained text embedding as the first layer and does **transfer learning** for the domain, which in this case is text classification of movie reviews. How well does the transfer learning work?

The model should achieve roughly 85% accuracy on binary classification. The important point of the exercise is gaining experience running the model.

Exercise 25.TFWV

Run a notebook such as `https://www.tensorflow.org/tutorials/text/word2vec` that learns word embeddings from a corpus using the skip-gram approach. Train the model on a corpus of Shakespeare, and separately on a corpus of contemporary language. Then test the resulting word embeddings on a downstream task such as classification of movie reviews. Does one word embedding model outperform the other?

Students should be able to show that the contemporary corpus does better than the Shakespeare corpus for downstream tasks involving contemporary language.

25.6 State of the art

Exercise 25.SOQA

Choose a dataset from https://paperswithcode.com/task/question-answering and report on the NLP Question-Answering model that performs best on that dataset. It will be easier if you choose a dataset for which both a paper and code are available. What made the highest-scoring model succesful and how does it compare to competing models?

The answer depends on the student's choice.

Exercise 25.SOGP

Experiment with a large-scale NLP text generation system. Pretrained online versions come and go; you could try https://6b.eleuther.ai/ or https://transformer.huggingface.co/doc/distil-gpt2 or search for another one. Give the system some prompts and evaluate the text it generates.

a. Is the generated text grammatical?

b. Is it relevant to the prompt's topic?

c. Is it internally consistent from sentence-to-sentence and paragraph-to-paragraph?

a. Usually the text is grammatical with only a few errors.

b. Usually it starts out relevant, but may wonder.

c. Consistency is not guaranteed. Character names can change; points may be repeated; actions may occur out of order.

Exercise 25.GPMM

Read https://medium.com/@melaniemitchell.me/can-gpt-3-make-analogies-16436605c446, Melanie Mitchell's account of trying to replicate her 1980s work on analogy-making with a standard GPT-3 model. Mitchell's 1980s program used symbolic AI to answer questions like "If a b c changes to a b d, what does i j k change to" with "i j l." In some cases GPT-3 can replicate this behavior, using only its pretrained model plus about four lines of prompt showing how the pattern of question and answer work. Experiment with a large-scale NLP text generation system to see how well it does with analogies using letters and numbers like this.

Students should be able to replicate some successes, and also note many failures. With numbers instead of letters, small numbers work well, but language models will do worse than humans at recognizing patterns such as squares and cubes of numbers.