Open Practical

[Chris Dyer, Phil Blunsom, Yannis Assael, Brendan Shillingford, Yishu Miao, Jan Buys]

The first three practicals covered a variety of introductory topics in Deep Learning for NLP. For the remainder of the term, you can select from among the following projects and explore them according to your interests. Treat the following project descriptions as suggested starting points, but feel free to take any of these ideas in a different direction if you feel inspired to. The practical sessions offer you an opportunity to pursue such ideas while making use of the support and expertise of your practical demonstrators.

Conditional language modeling

Conditioning on the topic set and generating a TED talk

This is the inverse of the problem considered in Practical 2/3: rather than conditioning on a talk, and predicting its labels, you should condition on an embedding of the label set and generate a TED talk.

This can be summarized by the very simple stochastic program:

$\mathbf{x}$ = embed(topic set) # map a set of topics into a (learned) vector representation

$\mathbf{y} \sim$ RNNLM($\mathbf{x}$) # feed the topic set into the RNN so it can use this information

The parameters of the RNN and the embedding model can be trained to maximize the log likelihood of a set of training pairs (topic set, $\mathbf{y}$)*.

You will need to design a function to create an embedding of the set of topics you are going to condition on. Options here are using an "RNN encoder" to read the list, an additive model, a convolutional architecture, or something else you come up with.

Questions

- Each topic is associated with an *unordered* set of topics. The simple additive embedding model obtains the same representation independent of the order of the elements of the set (because the commutative property holds for vector addition). What are the downsides of the additive model? On the other hand, the more sophisticated topic embedding models based on RNNs or convnets require presenting the topic labels in a particular order. Is this a problem? How might you address any problems you identify? Can you come up with a better embedding model?
- Using the same held-out data, how does the perplexity of your model compare to the perplexity of the **unconditional** language models proposed in Practical 3?
- In the Sutskever et al. (2014) paper we discussed in class, the initial states of the "decoder" LSTM were initialised using the input embedding ($\mathbf{x}$). There are alternatives: x can be

fed in as an input in addition to the embedding of the previous word at each timestep. Or it can be concatenated with the hidden state prior to projecting onto the output vocabulary. What are the expected advantages and disadvantages of these models? How well do these variants work in practice?

Conditioning on the talk and generating an summary

Summarisation is an important application in NLP. Sequence to sequence models are an appealing model for generating summaries: an encoder "reads" the contents of the talk and a decoder attends to the resulting representations and generates an output summary.

Each TED talk in our dataset comes with a short summary (located in the <description> XML tag). Design and implement a model for learning to generate summaries. Evaluate it using the ROUGE metric.

There are several challenges in this task:

- TED talks are quite long, and running a bidirectional RNN(/GRU/LSTM) over them is computationally expensive.
- Perhaps use a convolutional architecture instead. Convnets read windows of text "in parallel" so they can be much faster on vectorised hardware (like GPUs and modern CPUs).
- Perhaps process each sentence of the document in parallel with an RNN. This may require adapting the attention mechanism in some way. (Since you will no longer be attending to a position in a sequence, but perhaps to a position in a sequence of sequences.)
- Summaries often reuse content from the original document. In fact, there is a task called "extractive summarisation" that only lets you reuse existing content (this contrasts with the more general "abstractive summarisation"). Consider augmenting your summarisation model with a "copy mechanism" that decides to copy a word from a position, rather than encoding the word as a vector and re-decoding it.

Machine translation (MT)

Machine translation has been one of the notable successes of deep learning in NLP. The TED corpus has been translated into many languages by a volunteer effort. And TED now holds regional talks in many different languages, many of which are also translated into English. The result is: we have a lot of "in domain" training data to learn how to translate TED talks.

In this task, you should implement a sequence-to-sequence translation model for German to English translation, as described by Bahdanau et al. (2015). You can obtained preprocessed (tokenised, filtered for length, and lowercased, split into training/validation/test sets) parallel

German-English data (if you wish, you can copy this data somewhere here). Note: the validation/test sets contain OOV words relative to the vocabulary in the training set, in both the source and target languages. You will need to deal with this (the simplest thing is to replace infrequent words with an UNK token).

For decoding, you should implement two decoding algorithms: the greedy left-to-right decoder, and one that generates sample translations proportional to the probability p(translation | input). Evaluate the output of your translation model using multi-bleu.perl.

Questions

- Why does the greedy decoding algorithm make search errors even though it is possible to generate unbiased samples from the translation model?
- Sample 20 translations of the test set. What is the min/max/mean/median/stddev of the BLEU score? How do these compare to the BLEU score obtained by the greedy decoder?
- Do you expect word embeddings for English and German words that mean the same to be "close" in terms of cosine similarity? Why or why not?

Speaking rate modeling

How long does it take to read a document? Different speakers speak at different base rates; and different words take more or less time to pronounce.

In this task, you will use the timing data from the TED XML transcripts to build a model to predict how long different speakers will take to produce words. Since each speaker will have a different base rate, we will control for that by observing the speaking rate both in the training set and test set.

r = per-character speaking rate of speaker (observed in this case)

$\mathbf{h_t}$= context vector at time t (can be processed by token embedding, bidirectional RNN or convNet)

y_t = predict_duration $g(r, \mathbf{h_t})$ (in milliseconds? or log milliseconds? Should be positive?)

$loss = f(\mathbf{y}^*, \mathbf{y})$ (What loss function is used here? What does the error distribution look like? Is it symmetric? skewed?)

In this task, you will need to:

- Prepare the training data (e.g., compute r for each speaker in the training/validation/test splits)
- Decide what the regression target will be
- Compute the per-character speaking rate for every talk in the training/validation/test splits

- Design and implement the token embedding function (do you want to incorporate context or not?)
- Design the prediction function that converts each h_t to a scalar with the appropriate range (e.g., strictly positive)
- Design the loss function.

Questions

- Use a character-based embedding function to represent each token (i.e., compose a sequence of characters into a "word embedding" using a convnet or RNN). What are the advantages and disadvantages this of modeling the speaking rates of English words? Which model works better?
- This model is only trained to predict the length of the full sequences. The lengths of individual words are not present in the training data. Plot the durations of words predicted by your model on the validation set. Do these seem reasonable? How might you improve the training objective to give better per-word predictions?

Propose your own task

If none of the above tasks inspire you, or if you happen to have discovered an interesting source of data to model, feel free to propose and pursue your own research idea.

Some ideas might be:

- Using the multilingual data from the MT task, design and implement a word embedding model such that words that have a similar meaning in English (e.g.: reject, deny) have vectors that are similar to each other, and words that are similar in meaning in translation have vectors that are close to each other (e.g., reject, ablehnen).
- Do something with the audio/video recordings and the transcripts (e.g., train an ASR system based on conditional language models).
- When will the audience laugh? The TED talk contain indications for when the audience laughs. Can you predict what's going to cause the audience to laugh using the text?