Natural Language Understanding
Spring 2017

Machine Learning Institute
Dept. of Computer Science, ETH Zürich
Prof. Dr. Thomas Hofmann
Dr. Massimiliano Ciaramita
Web http://www.da.inf.ethz.ch/teaching/2017/NLP/

Project Description

Before sending us questions regarding the project, please first see the FAQ at

http://da.inf.ethz.ch/teaching/2017/NLP/project.php

Always email both of us, Jason and Florian.

1 Task 1: RNN Language Modelling (30+10 Points)

1.1 1a) Language Modelling (30 Points)

Your task is to build a simple LSTM language model. To be precise, we assume that words are independent given the recurrent hidden state, we compute a new hidden state given the last hidden state and last word, and predict the next word given the hidden state:

$$P(w_1, \ldots, w_n) = \prod_{t=1}^{n} P(w_t|\mathbf{h}_t)$$
$$P(w_t|\mathbf{h}_t) = \mathrm{softmax}(\mathbf{W}\mathbf{h}_t)$$
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, w^\star_{t-1})$$

where f is the LSTM recurrent function, $\mathbf{W} \in \mathbb{R}^{|V| \times d}$ are softmax weights and $\mathbf{h}_0$ is either an all-zero constant or a trainable parameter.

You can use the tensorflow `cell` implementation [1] to carry out the recurrent computation in f. However, you must construct the actual RNN yourself (e.g. don't use tensorflow's or any other RNN library). That means, you will need to use a python loop that sets up the unrolled graph. To make your life simpler, please follow these design choices:

Model and Data specification

- Use a special sentence-beginning symbol <bos> and a sentence-end symbol <eos> (please use exactly these, including brackets). The <bos> symbol is the input, when predicting the first word and the <eos> symbol you require your model to predict at the end of every sentence.
- Use a maximum sentence length of 30 (including the <bos> abd <eos> symbol). Ignore longer sentences during training and testing.
- Use a special padding symbol <pad> (please use exactly this, including brackets) to fill up sentences of length shorter than 30. This way, all your input will have the same size.
- Use a vocabulary consisting of the 20K most frequent words in the training set, including the symbols <bos>, <eos>, <pad> and <unk>. Replace out-of-vocabulary words with the <unk> symbol before feeding them into your network (don't change the file content).
- Provide the *ground truth* last word as input to the RNN, not the last word you predicted. This is common practice.
- Language models are usually trained to minimize the cross-entropy. Use tensorflow's

`tf.nn.sparse_softmax_cross_entropy_with_logits`

to compute the loss[1]. Use the AdamOptimizer to minimize the loss. Use `tf.clip_by_global_norm` to clip the norm of the gradients to 10.

- Use a batch size of 64.
- Use the data at [6]. Don't pre-process the input further. All the data is already white-space tokenized and lower-cased. one sentence per line.
- To initialize your weight matrices, use the `tf.contrib.layers.xavier_initializer()` initializer introduced in [5].

Experiments All experiments should not run for longer than, say, four hours on the GPU. For this task, your grade won't improve with performance.

- **Experiment A**: Train your model with word-embedding dimensionality of 100 and a hidden state size of 512 and compute sentence perplexity on the evaluation set (see submission format below).
- **Experiment B**: It is common practice, to pretrain wordembeddings using e.g. word2vec. This should make your model train faster as words will come already with a useful representation. Use the code at [3] to load these word embeddings [4] trained on the same corpus. Train your model again and compute evaluation perplexity.
- **Experiment C** It is often desirable to make the LSTM more powerful, by increasing the hidden dimensionality. However, this will naturally increase the parameters $\mathbf{W}$ of the softmax. As a compromise, one can use a larger hidden state, but down-project it before the softmax. Increase the hidden state dimensionality from 512 to 1024, but down-project h_t to dimensionality 512 before predicting w_t as in

$$\tilde{\mathbf{h}}_t = \mathbf{W}_P \mathbf{h}_t$$

where $\mathbf{W}_P$ are parameters. Train your model again and compute evaluation perplexity.

Submission and grading

- Grading scheme: 100% correctness.
- Deadline May 7th, 23:59:59.
- You are not allowed to copy-paste any larger code blocks from existing implementations.
- Hand in
  - Your python code
  - **Three** result files containing sentence-level perplexity numbers on the **test** set (to be distributed) for all three experiments. Recall that perplexity of a sentence $S = \langle w_1, \ldots, w_n \rangle$ with respect to your model $p(w_t|w_1, \ldots, w_{t-1})$ is defined as

$$\text{Perp} = 2^{-\frac{1}{n}\sum_{t=1}^{n} \log p(w_t|w_1,\ldots,w_{t-1})}$$

    The <eos> symbol is part of the sequence, while the <pad> symbols (if any) are not.

    Input format sentences.test
    One sentence (none of them is longer than 28 tokens) per line:

```
beside her , jamie bounced on his seat .
i looked and saw claire montgomery looking up at me .
people might not know who alex was , but they knew to listen to him .
```

[1] This operation *fuses* the computation of the soft-max and the cross entropy loss given the logits. For numerical stability, it's very important to use this function.

Required output format groupXX.perplexityY
(where XX is your **group number** and Y$\in$ {A,B,C} is the experiment). One perplexity number per line:

```
10.232
2.434
5.232
```

1.2 Conditional Generation (10 Points)

Let's use your trained language model from above to generate sentences. Given an initial sequence of words, your are asked to **greedily** generate words until either your model decides to finish the sentence (it generated <eos>) or a given maximum length has been reached. Note, that this task does not involve any training. Please see the tensorflow documentation on how to save and restore your model from above.

There are several ways how to implement the generation. For example, you can define a graph that computes just one step of the RNN given the last input and the last state (both from a new placeholder).

$$\text{state}_t, p_t = f(\text{state}_{t-1}, w_{t-1})$$

That means, for a prefix of size m and a desired length of n, you run this graph n times. The first $m+1$ times you take the input form the prefix. For the rest of the sequence, you take the most likely[2] word $w^{t-1} = \text{argmax}_w\, p_{t-1}(w)$ from the last step.

- Grading scheme: 100% correctness.
- Deadline May 7th, 23:59:59.
- You are not allowed to copy-paste any larger code blocks from existing implementations.
- Hand in
  - Your python code
  - Your continued sentences of length up to 20. Use your trained model from experiment **C** in task 1.1.

    Input format sentences.continuation One sentence (of length less than 20) per line:

    ```
    beside her ,
    i
    people might not know
    ```

    The <bos> symbol is not explicitly in the file, but you should still use it as the first input.

    Required output format groupXX.continuation (where XX is your **group number**)

    ```
    beside her , something happened ! <eos>
    i do n't recall making a noise , but i must have , because bob just looked up from his
    people might not know the answer . <eos>
    ```

[2] You can compute the argmax in python or in tensorflow.

2 Task 2: Dialogue (60 Points)

Your task is to build a simple dialogue system, or conversational agent if you want, that can handle real-world conversations. Starting from a simple seq2seq [7] implementation, we ask you to identify at least one limitation of the baseline and propose, implement and evaluate an extension that tackles the identified problem. Your approach does not have to be novel in the literature.

Model and Data specification

- Use the triple dataset as input (link in the email). While the dataset contains conversation triples of the form

  " volunteer duty ". | i didn ' t volunteer . | you causing trouble again ?

  you should treat them as two tuples

  " volunteer duty ". | i didn ' t volunteer .
  i didn ' t volunteer . | you causing trouble again ?

  each, unless you want to explicitly exploit the triple-nature in your model.

  For licensing reasons: **Please, do not distribute the dataset!**

- Use a seq2seq model without any additional architecture tweak (e.g. attention) as your baseline.
- You are free to choose any extension of the baseline as your proposed model (see below).

Submission and grading

- Grading scheme: 50% creativity of your extension 30% writeup 20% performance.
- Deadline June 9th, 23:59:59.
- You are allowed to use any existing code but **you must cite** the source and explain which parts you have written yourself.
- Hand in
  - Your python code
  - Your model (not the baseline) that can be run in test mode. Please submit a link to a zipped directory at polybox. The directory needs to include a script `run-test.sh` that can take a filename as input[3] to produce an output containing two numbers per line. The first number is the perplexity of the second turn (e.g. *i didn ' t volunteer .*) given the first turn and the second number is the perplexity of the third turn (e.g. *you causing trouble again ?*) given the second (and possibly first) turn. As an example,

    ```
    ./run-test.sh some-triples.txt
    ```

    prints

    ```
    232.23 482
    32.9 43
    45.2 150
    ...
    ```

    as output. Of course, `run-test.sh` should only run testing, not training. That means, the zipped directory will need to contain your saved models, which is why you should upload it to polybox and not send it as an attachment.
  - A **two to four** page report written in LaTeX that describes your model. Your report should reflect the motivation of your approach, contain a precise description of it, possibly reference to existing literature, and compare your results to the baseline (qualitatively and quantitatively). You might want to include sample input-output pairs to evaluate your model qualitatively, which is why the number of pages can be up to four.

[3] The input file is of the same format as `Validation_Shuffled_Dataset.txt`

Recommended steps

- Read the original seq2seq paper[7].
- Implement the baseline. Do not hesitate to google for an existing implementation. We don't grade your coding effort.
- Identify a problem, common patterns of failure or additional data that you think would help.
- You might want to do a quick literature search on possible extensions. Nevertheless, we provide you with some possible directions (we don't say those are equally complicated or mutually exclusive):

  1. Use the three-turn structure of the dataset, e.g. Use the first turn when predicting the third.
  2. Use the Cornell Movie Corpus as additional training data[10].
  3. Attention mechanism[16].
  4. Maintaining a specific model of the utterer for better consistency[11].
  5. A different objective function for seq2seq for generating diverse responses[12].
  6. Latent variable model to capture the topic of conversation[14].
  7. Copying mechanism for seq2seq to combat UNK issues[15].
  8. Reinforcement Learning-based approach[13].

  Caveat: Implementing some of these suggestions might be quite involved, so we recommend to stick to something relatively simple, at least initially.

Infrastructure

You must use tensorflow but any programming language is allowed. However, we strongly recommend python3. You have access to two compute resources: Unlimited CPU usage on euler and limited GPU usage on Azure. Note that the difference in speed is typically a factor between 10 and 100. Do any debugging on euler and use your GPU hours wisely once you are relatively sure that your model is bug-free.

2.1 Running on Azure

Machine setup In your group email address you should find a link that allows you to access the Azure group portal. Under [8] you can create a new machine from a template by clicking 'edit'. Copy-paste the json code from [9], click 'save', enter username and password and click 'purchase'. After about one hour, the machine will show up under 'virtual machines' as GpuVM. There will also be another machine called TransferVM. Please stop this one after an hour – otherwise it will cost you money. Click on GpuVM to see it's IP address and ssh onto it.

Tensorflow setup To run python3 with tensorflow, please follow these steps

- Add

  ```
  export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/local/cuda/lib64:/usr/local/cuda/extras/CUPTI/lib64"
  export CUDA_HOME=/usr/local/cuda
  ```

  at the end of your `~/.bashrc` file

- Run

  ```
  sudo apt-get upgrade nvidia-settings
  sudo reboot
  ```

  Once the machine is booted again, tensorflow 1.0.0 will be available.

Pricing Every machine running costs 1.60$ per hour, no matter what it does e.g. whether it is on idle or full load. We hope you can also see a summary at https://www.microsoftazuresponsorships.com/.

2.2 Running on Euler

Please see the wiki on how to use euler, in particular on how to request certain amounts of memory and compute power [2].

Run

```
module load new gcc/4.8.2 python/3.6.0
```

to get a running python tensorflow implementation. Before you allocate dozens of cores, use `bjob_connect` and `top` to investigate how many cores tensorflow is actually using. Often this not more than four. In any case you **must** set `inter_op_parallelism_threads` and `intra_op_parallelism_threads` in tensorflow to match the number of cores that you ordered when submitting jobs. The admins keep an eye on this.

References

[1] https://www.tensorflow.org/versions/r0.12/api_docs/python/rnn_cell/rnn_cells_for_use_with_tensorflow_s_core_rnn_methods

[2] http://brutuswiki.ethz.ch/brutus/EULER_for_beta_users#Useful_bsub_options

[3] http://da.inf.ethz.ch/teaching/2017/NLP/material/load_embeddings.py

[4] https://polybox.ethz.ch/index.php/s/cpicEJeC2G4tq9U

[5] http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

[6] https://polybox.ethz.ch/index.php/s/qUc2NvUh2eONfEB

[7] https://arxiv.org/abs/1409.3215

[8] https://portal.azure.com/#create/Microsoft.Template

[9] https://polybox.ethz.ch/index.php/s/6gdlnBdi7jboOmr

[10] https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html

[11] https://arxiv.org/pdf/1603.06155.pdf

[12] https://arxiv.org/pdf/1510.03055.pdf

[13] https://arxiv.org/pdf/1606.01541.pdf

[14] https://arxiv.org/pdf/1605.06069.pdf

[15] https://arxiv.org/pdf/1603.06393.pdf

[16] https://arxiv.org/pdf/1409.0473.pdf