SUMO's new bAbI: A Neuro-Symbolic Approach to Natural Language Inference

Adam Pease and Josef Urban

We are conducting experiments in machine translation from natural language to logic using SUMO `https://github.com/ontologyportal/sumo` terms. The code to create the datasets for training is in `com.articulate.sigma.mlpipeline.GenSimpTestData` at `https://github.com/ontologyportal/sigmakee`.

The ML training resource used is https://github.com/tensorflow/nmt

This SigmaKEE code generates language-logic pairs designed for training a machine learning system. Several approaches are used:

- instantiate relations with arguments of appropriate types and then generate NL paraphrases from them
- run through all formulas in SUMO and generate NL paraphrases
- build up sentences and logic expressions compositionally

The compositional generation is potentially the most comprehensive. It consists of building ever more complex statements that wrap or extend simpler statements. Originally, this meant starting with a simple subject-verb-object construction and adding:

- indirect objects
- tenses for the verbs
- modals

Ultimately we believe we can feed natural text to the system and save formulas that pass test for correctness from SigmaKEE

We should be able to wrap everything in negated/not-negated, likely/unlikely.

We plan to trap bad constructions by using language models to eliminate rare combinations of words, maybe.

1 Appendix: Mechanics

I put the following in my .bashrc

```
/home/mptp/miniconda3/condabin/conda init bash
source /home/mptp/.bashrc
conda activate tf_gpu112
```

I run the following to produce the SUMO-based training files

```
$ time java -Xmx14g -classpath /home/apease/workspace/sigmakee/lib/*:/home/apease/workspa

$ time java -Xmx60g -classpath /home/apease/workspace/sigmakee/lib/*:/home/apease/workspa

$ time java -Xmx14g -classpath /home/apease/workspace/sigmakee/lib/*:/home/apease/workspa
```

Each command produces a -log.txt and -eng.txt file. Make sure they completed and each file is the same number of lines as its sibling. Then concatenate them all together

```
$ cat allAxioms-eng.txt groundRelations-eng.txt outKindaSmall-eng.txt > combined-eng.txt
$ cat allAxioms-log.txt groundRelations-log.txt outKindaSmall-log.txt > combined-log.txt
```

The two -eng and -log files can go anywhere. They first need some pre-processing with

```
$ ~/workspace/sumonlp/preprocess.sh ~/nfs/data/combined combined
```

Note that it produces the vocab.* files that will typically have empty (or just whitespace) lines in them. You need to delete these empty lines manually (or improve the script).

The result will be files call trainX, devX and testX where X is ".nat" and ".sum1" in the "combined" directory. When running the training process the script needs to be run from a directory

where you have installed nmt. For me that's in /test1

```
$ ~/workspace/sumonlp/training.sh ~/nfs/data/combined
```

Then you can test it

```
$ ~/workspace/sumonlp/test.sh ~/nfs/data/combined/models/model
```

Note that you should first check which GPU cards are free by runing

```
nvidia-smi
```

When you see that e.g. GPU number 1 is free, do the following to put your job there:

```
export CUDA\_VISIBLE_DEVICES=2
```

Note that this number seems to be off by 1 on air-05 (likely a peculiarity of air-05).

If you are getting OOM (Out of Memorey) crashes, decrease the batch size in the training script in this line:

```
  --batch_size=768 \
```

e.g.
to 32 (and fine tune it later if its too low/high).

Don't forget to delete the model directory if you change such parameters, otherwise the old params will be kept and read from the file hparams there.

Running many of these steps can take time. Here are some current values to support expectations and estimates

- We can generate about 1M language-logic pairs per hour on my laptop (GenSimpleTestData -s). as of Dec 6
- generating allAxioms (GenSimpleTestData -a) real 2m6.851s (as of November)
- generating all ground statements (GenSimpleTestData -g) real 5m58.213s (as of November)
- on a 1.5 GB file, running preprocess.sh takes - real 9m34.599s (as of November)
- training.sh takes - real 3m31.257s (as of November)
- test.sh takes - real 3m32.435s (as of November)
- six tenses (part, present, future and progressive or not) and correspond to SUMO expressions relative to a diectic 'Now'
- negation (for action sentences and separately for attitude wrappers - believes, knows etc)
- frequency-based selection of nouns and verbs
- human names and social roles
- restrict human actions with CaseRole of 'agent' to be IntentionalProcess(es) and otherwise use the CaseRole of 'experiencer'
- use WordNet's noun and verb morphological exception lists
- counts of objects (0-10) are used randomly a small portion of the time, divided between using numerals some times and digits other times
- quantities of substances with units are used randomly a small portion of the time
- Some sentences have metric dates and times, in several formats
- Queries ('what','who') are generated sometimes, and with a question mark
- WordNet verb frames are used that index verbs to a context of use. However, this is quite limited compared to VerbNet and FrameNet, and still allows many semantically non-sensical sentences to be generated, while also missing some common sentence forms, especially use of prepositions
- Sentences with all variations of SUMO's modalAttribute are generated (Possibility, Obligation etc)

- Sentences with all variations of SUMO's PropositionalAttitudes are generated 'believes','knows','says' etc
- Sentences with 'says' have their content quoted
- Imperative sentences with the English "You (understood)" form

References