[Current Next Steps 2](#_Toc51586811)

[University side 2](#_Toc51586812)

[Ordbogen side 4](#_Toc51586813)

[Preliminary information (Datasets, graphs) 5](#_Toc51586814)

[SemCor – current 5](#_Toc51586815)

[SemCor stats (80-10-10, lowercased) 5](#_Toc51586816)

[Graph (min\_freq=2, with dummySense) 5](#_Toc51586817)

[LM\_TransformerXL 6](#_Toc51586818)

[Mini-experiments and checks 7](#_Toc51586819)

[Simple GRUs 7](#_Toc51586820)

[With freezing 7](#_Toc51586821)

[Part 1 8](#_Toc51586822)

[WikiText-2 (standard LM) 8](#_Toc51586823)

[Experiment 8](#_Toc51586824)

[2 GRUs on SemCor 9](#_Toc51586825)

[Simple GRUs 9](#_Toc51586826)

[GNN Input Signal 11](#_Toc51586827)

[2GRUs + GAT 11](#_Toc51586828)

[2GRUs based only on GAT, concat\_input\_dim=300 12](#_Toc51586829)

[SelectK 13](#_Toc51586830)

[Reinserting the SelectK method 13](#_Toc51586831)

[Checking code & output 14](#_Toc51586832)

[K=1, selecting the logits from the senses’ RNN 16](#_Toc51586833)

[K=10, selecting the logits from the senses’ RNN 17](#_Toc51586834)

[K=1 – with freezing – selecting the logits from the senses’ RNN 17](#_Toc51586835)

[K=10 – with freezing – selecting the logits from the senses’ RNN 18](#_Toc51586836)

# Current Next Steps

## University side

Step 1: re-run the pipeline for SemCor, and try to replicate exactly the old result with 185 Valid-PPL. This would demonstrate that the architecture still works as intended.

Status: done,

the old result was so good because the SemCor corpus had been **lowercased**, and 2/3/n digit numbers had been turned into <num>.

we got 219.4 because we used an LSTM where a GRU works better. See [Experiment](#_Experiment).

This validates the architecture (i.e. it’s not bugged, as we also see from WikiText-2). However, when we try experiments on SemCor’s 650K training tokens, we get very high PPL values, like 445 Valid-PPL at [Experiment C](#_Experiment_C).

We recall that Penn-Treebank has 929K training tokens with heavy pre-processing, namely “…lower-cased, numbers replaced with N, punctuation removed. The vocabulary is the most frequent 10k words”. WikiText-2 has ~2 million training tokens.

Since the RNN architecture works as intended and is not bugged, we try to reach better performance by including *more* sense-labeled data, see Step 3.

Step 2: try a 1024>1024>512 architecture on WT-2, with no dropout, to try to bring the Valid-PPL well under 200.

status:

done. The first experiment on WT-2 is the one with a 1150>1150>400 LSTM that (mistakenly) applies a standard dropout of p=0.1. See [Experiment A](#_Experiment_A), it reaches 200.5 Validation PPL.

Moreover, [Experiment B](#_Experiment_B), using a 1024>1024>512 GRU with no dropout, reaches **186.4** Valid-PPL

Step 3: gather enough sense-labeled data to obtain a corpus of size comparable to WT-2, to try to obtain a similar Valid-PPL.

I need to review the other Sense-Labeled Corpuses. The ones eligible for inclusion are those labelled with **WordNet 3.0** senses, i.e.:

SemCor (228k annotated / 778k words), OMSTI (1 million / 35 millions), MASC (115K / 596K). Ontonotes would be too, but they are saved in a complex acess-DB format. + SemEval 2013 (8.3K words), SemEval 2015(2.6K)

The current expanded SLC contains SemCor + MASC + the first 300 MB of text of OMSTI (for the sake of speed in the current experiments, we did not include it in its entirety).

training tokens: 4,875,767. Validation tokens: 605,366

The first tentative [Exp A](#_Exp_A) reaches 158 Valid-PPL, but there is a chance I am reading one of the subcorpuses twice so it may be bugged. It must be checked and re-done.

Step 4: continue adjusting the code and running mini-experiments, to prepare the freezing mechanism.

Using the flags predict\_senses=**True** in training\_setup() and with\_freezing=**True** in training\_loop(), we apply the current version of the freezing mechanism:

* we set p.requires\_grad=**False** on all the parameters of the globals’ GRU and softmax
* We recreate the optimizer, to eliminate any remaining momentum
* We activate the senses’ GRU with model\_forParameters.predict\_senses = **True**.
* We set after\_freezing\_flag = **True**, so the loss will be computed as   
  loss = loss\_sense instead of loss = loss\_global + loss\_sense

Despite all of this, the globals’ loss still changes in the epochs after the freezing point.

ToDo:

build a small toy-model, that takes in the same input and labels, and examine the freezing mechanism.

Step 5: Finish the freezing mechanism – a very minor oscillation due to the embeddings is still ok

status: done. The freezing mechanism now works, and without any oscillation, as we can see in [Full RNN (RNNs, embeddings, GATs)](#_Full_RNN_(RNNs,).

Step 6: Use lowercased SemCor, for the sake of brevity and development. Execute Senses-with-RNN experiments, with and without the freezing mechanism

Step 7a:

Set up and run, indicatively, the following experiments:

* On SemCor alone, and on pretrained on WT-2+SemCor
* With and without freezing
* With and without GAT-global-nodestate
* Standard GRU for globals and senses
* SelectK(1,5,10,50), with and without freezing
* Attention mechanism for senses

This would mean a total of: 2x2x2x3 = 24 experiments

(alternatives that work better get priority when it comes to computational resources)

Step 7b:

Start the write-up of the paper. Proceed with the structure of the experiments and the description of the methods, and also abstract + background

## Ordbogen side

Information 1:

The Desktop version is currently put aside and not developed. The version currently developed is the web browser add-on version.

The GrammaTip team is also working on a text editor.

Information 2:

You can (ideally) reliably convert to & fro Pytorch and Tensorflow (see instrument: Neural Network Exchange)

Information 3:

Instrument for deployment that could be used: MLFlow.

Task 4:

Look at the transformer-LM code. Can we train it as expected? Can we operate on both Danish and English?

# Preliminary information (Datasets, graphs)

## SemCor – current

### SemCor stats (80-10-10, lowercased)

Note: currently I am also including punctuation in the vocabulary from SLC.

Training tokens: 646,038

Validation tokens: 80,760

|Vocabulary|= 22,782

### Graph (min\_freq=2, with dummySense)

INFO : Constructing X, matrix of node features

INFO : X\_senses.shape=torch.Size([**41206**, 200]) # senses with data: 30445

INFO : X\_globals.shape=torch.Size([**22782**, 200])

INFO : X\_definitions.shape=torch.Size([30446, 200])

INFO : X\_examples.shape=torch.Size([27921, 200])

INFO : E\_embeddings.shape=torch.Size([22782, 200])

Graph ranges: [0, senses, 41206, globals, 63988, defs, 94434, examples, 122355]

INFO : Defining the edges: def, exs

INFO : def\_edges\_se.\_\_len\_\_()=30446

INFO : exs\_edges\_se.\_\_len\_\_()=27921

INFO : Defining the edges: sc

INFO : Reading the sense-labeled corpus, to create the connections between globals and the senses that belong to other words.

INFO : Generator over subcorpus at TextCorpuses/My Sense-Labeled Corpus/Training/semcor.xml

INFO : sc\_edges\_with\_external.\_\_len\_\_()=7044

INFO : sc\_edges.\_\_len\_\_()=48250

INFO : get\_edges\_selfloops>max\_sense=41205

INFO : []

INFO : len(edges\_ls)==0

INFO : sc\_edges\_with\_selfloops.\_\_len\_\_()= 48250

INFO : Defining the edges: syn, ant

INFO : syn\_edges.\_\_len\_\_()= 19804 (fewer, it was 24272…(?)(is this relevant or only a modification from an older version)

INFO : ant\_edges.\_\_len\_\_()=4070

INFO : Pre-computing and saving graphArea matrix, with area\_size=32

~~D~~ata(edge\_index=[2, **130491**], edge\_type=[130491], node\_types=[122355], num\_relations=[1], x=[**122355**, 100]) ; file size =~151MB

# LM\_TransformerXL

# Mini-experiments and checks

This section is for the mini-experiment to check the functionality of the various architectures

## Simple GRUs

### With freezing

INFO : RNN(

(dropout): Dropout(p=0, inplace=False)

(main\_rnn\_ls): ModuleList(

(0): GRU(300, 1024)

(1): GRU(1024, 1024)

(2): GRU(1024, 512)

)

(senses\_rnn\_ls): ModuleList(

(0): GRU(300, 1024)

(1): GRU(1024, 1024)

(2): GRU(1024, 512)

)

(linear2global): Linear(in\_features=512, out\_features=22782, bias=True)

(linear2senses): Linear(in\_features=512, out\_features=41206, bias=True)

)

INFO : Parameters:

INFO : ('E', torch.Size([22782, 300]), torch.float32, True)

('memory\_hn', torch.Size([3, 2, 1024]), torch.float32, False)

('memory\_hn\_senses', torch.Size([3, 2, 1024]), torch.float32, False)

('main\_rnn\_ls.0.weight\_ih\_l0', torch.Size([3072, 300]), torch.float32, True)

…

('linear2global.bias', torch.Size([22782]), torch.float32, True)

('linear2senses.weight', torch.Size([41206, 512]), torch.float32, True)

('linear2senses.bias', torch.Size([41206]), torch.float32, True)

INFO : Number of trainable parameters=65.13M

|  |  |  |  |
| --- | --- | --- | --- |
| *Epoch* | *Training* | | |
|  | Globals | Senses | Multi-senses |
| 1 | 22831.58 | 1.0 | 1.0 |
| … |  |  |  |
| 5 | 8954.96 | 1.0 | 1.0 |
| 6 | 3708.13 | 40549.28 | 41154.67 |
| 7 | **3702.04** | 37411.89 | 39028.87 |
| 8 | **3702.04** | 33370.31 | 36221.34 |

## SelectK

### With freezing

INFO : Model:

INFO : SelectK(

(dropout): Dropout(p=0, inplace=False)

(main\_rnn\_ls): ModuleList(

(0): GRU(300, 1024)

(1): GRU(1024, 1024)

(2): GRU(1024, 512)

)

(senses\_rnn\_ls): ModuleList(

(0): GRU(300, 1024)

(1): GRU(1024, 1024)

(2): GRU(1024, 512)

)

(linear2global): Linear(in\_features=512, out\_features=22782, bias=True)

(linear2senses): Linear(in\_features=512, out\_features=41206, bias=True)

)

INFO : Number of trainable parameters=65.13M

model.K=1

|  |  |  |  |
| --- | --- | --- | --- |
| *Epoch* | *Training* | | |
|  | Globals | Senses | Multi-senses |
| 1 | 22831.58 | 1.0 | 1.0 |
| … |  |  |  |
| 5 | 8954.96 | 1.0 | 1.0 |
| 6 | 3708.13 | 39810810.26 | 100000092.9 |
| 7 | **3702.04** | 39810810.26 | 100000092.9 |
| 8 | **3702.04** | 39810810.26 | 100000092.9 |

# Part 1

## WikiText-2 (standard LM)

Standard Language Model task, i.e. globals only.

Objective: train a model on WikiText-2, that will be stored and used as a pretrained base to operate on SemCor.

### Experiment

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Input Signals** | **Hyperparameters** |
| Globals: **Main GRU** with 3 layers (1024>1024>512) | 1) The word embedding of the current global (d=300) | batch\_size=40  TBPTT length=35 |
| Senses:nothing. We create a GRU, but we do not use it. | grapharea=32, hops=1  learning rate=0.00005 |
| *Later on we may also add other mechanisms and parameters to the model object.* |  |

Number of trainable parameters=78.65M. Expecting a Valid-PPL in the range 185-190

|  |  |  |
| --- | --- | --- |
| *Epoch* | *Training PPL, globals* | *Validation PPL, globals* |
| 1 | 1435.47 | 704.11 |
| 2 | 676.57 | 445.01 |
| 3 | 481.45 | 358.82 |
| 4 | 391.48 | 313.99 |
| 5 | 336.84 | 285.54 |
| 6 | 298.43 | 265.36 |
| 7 | 269.1 | 250.15 |
| 8 | 245.64 | 238.23 |
| 9 | 226.35 | 228.74 |
| 10 | 210.11 | 221.11 |
| 11 | 196.2 | 214.92 |
| 12 | 184.1 | 209.79 |
| 13 | 173.43 | 205.52 |
| 14 | 163.9 | 201.92 |
| 15 | 155.32 | 198.86 |
| 16 | 147.54 | 196.2 |
| 17 | 140.42 | 193.93 |
| 18 | 133.87 | 192.02 |
| 19 | 127.83 | 190.42 |
| 20 | 122.22 | 189.08 |
| 21 | 116.99 | 188.01 |
| 22 | 112.11 | 187.16 |
| 23 | 107.52 | 186.52 |
| 24 | 103.21 | 186.07 |
| 25 | 99.14 | 185.74 |
| 26 | 95.3 | 185.57 |
| 27 | 91.65 | **185.53**  'correct\_g': 47667,  'top\_k\_g': 104562, 'tot\_g': 212800 |
| 28 | 88.2 | 185.68 |

We saved the model @epoch 28. We store it, to be used later as a pretrained model.

What could be the next steps?

* Using 2 simple GRUs on SemCor, providing a baseline for the sense prediction
* Adding the GAT nodestate of the global should provide additional information, and ideally improve both the standard LM and the senses’ prediction.  
  From this, 2 experiments would follow: Simple GRUs + GAT input, with and without freezing.
* Using the pretrained WT-2 model? This should improve the performance of the standard LM task on globals…  
  However, it is likely that using the pretrained WT-2 is more significant on methods that actually rely on the globals’ performance (SelectK).
* Then, moving on to the other mechanisms

## 2 GRUs on SemCor

### Simple GRUs

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Input Signals** | **Hyperparameters** |
| Globals: **Main GRU** with 3 layers (1024>1024>512) | 1) The word embedding of the current global (d=300) | batch\_size=32  TBPTT length=35 |
| Senses: **Senses’** **GRU** with 3 layers, as above. |  | grapharea=32, hops=1  learning rate=0.00005 |

INFO : Model:

INFO : DataParallel(

(module): RNN(

(dropout): Dropout(p=0, inplace=False)

(main\_rnn\_ls): ModuleList(

(0): GRU(300, 1024)

(1): GRU(1024, 1024)

(2): GRU(1024, 512)

)

(senses\_rnn\_ls): ModuleList(

(0): GRU(300, 1024)

(1): GRU(1024, 1024)

(2): GRU(1024, 512)

)

(linear2global): Linear(in\_features=512, out\_features=24122, bias=True)

(linear2senses): Linear(in\_features=512, out\_features=41534, bias=True)

))

Number of trainable parameters=65.13M,

where embeddings=22782x300=6.84M, softmax=22782x513+41206x513=32.83M , core=25.46M (i.e. 12.73M x 2)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Epoch* | *Training* | | | *Validation* | | |
|  | Globals | Senses | Multi-senses | Globals | Senses | Multi-senses |
| 1 | 624.45 | 943.69 | 11541.31 | 507.95 | 646.48 | 11236.47 |
| 2 | 358.54 | 593.88 | 8199.44 | 403.42 | **632.07** | 10391.08 |
| 3 | 291.33 | 586.87 | 8037.58 | 327.9 | 636.47 | 10305.85 |
| … |  |  |  |  |  |  |
| 17 | 114.24 | 547.9 | 7021.49 | **224.22** | 697.34 | 11324.48 |
| 18 | 110.53 | 529.92 | 6545.85 | 224.28 | 739.26 | 14473.89 |
| 19 | 107.04 | 517.93 | 6322.95 | 224.55 | 719.44 | 12236.07 |

## GNN Input Signal

### 2GRUs + GAT

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Input Signals** | **Hyperparameters** |
| Globals: **Main GRU** with 3 layers (1024>1024>512) | 1) The word embedding of the current global *wi* (d=300)  2) The state vector of the global node *I*, obtained applying a GNN (GAT) on the dictionary graph. (d=**200**). | batch\_size=**32**  TBPTT length=35 |
| Senses: **GRU** with 3 layers, as above. | grapharea=32, hops=1  learning rate=0.00005 |
| **GNN**: GAT, with input\_d=200 and output=100x2 |

INFO : DataParallel(

(module): RNN(

(dropout): Dropout(p=0, inplace=False)

(gat\_globals): GATConv(200, 100, heads=2)

(main\_rnn\_ls): ModuleList(

(0): GRU(500, 1024) (1): GRU(1024, 1024) (2): GRU(1024, 512) )

(senses\_rnn\_ls): ModuleList(

(0): GRU(500, 1024) (1): GRU(1024, 1024) (2): GRU(1024, 512) )

(linear2global): Linear(in\_features=512, out\_features=22782, bias=True)

(linear2senses): Linear(in\_features=512, out\_features=41206, bias=True) ))

N. of parameters=90.87M

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Epoch* | *Training* | | | *Validation* | | |
|  | Globals | Senses | Multi-senses | Globals | Senses | Multi-senses |
| 1 | 598.7 | 927.2 | 11685.38 | 428.84 | 645.24 | 11085.52 |
| 2 | 323.63 | 593.62 | 8195.95 | 355.01 | **630.39** | 10221.31 |
| 3 | 262.49 | 587.24 | 8052.54 | 309.42 | 636.85 | 10305.34 |
| … |  |  |  |  |  |  |
| 17 | 112.68 | 559.39 | 7334.36 | **223.8** | 708.85 | 11857.94 |
| 18 | 109.13 | 545.7 | 6959.92 | 223.89 | 711.23 | 11795.71 |
| 19 | 105.78 | 533.85 | 6683.94 | 224.17 | 716.01 | 12018.15 |

Valid PPL:

### 2GRUs *based only on* GAT, d=150x2

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Input Signals** | **Hyperparameters** |
| Globals: **Main GRU** with 3 layers (1024>1024>512) | 1) The state vector of the global node *I*, obtained applying a GNN (GAT) on the dictionary graph. (d=150). | batch\_size=**32**  TBPTT length=35 |
| Senses: **GRU** with 3 layers, as above. | grapharea=32, hops=1  learning rate=0.00005 |
| **GNN**: GAT, input\_d=150, output=150x2 |

Parameters=77.52M

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Epoch* | *Training* | | | *Validation* | | |
|  | Globals | Senses | Multi-senses | Globals | Senses | Multi-senses |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 18 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 22 |  |  |  |  |  |  |
| 23 |  |  |  |  |  |  |

# SelectK

## Reinserting the SelectK method

We paste the latest version of the GRU in the Senses.py module, in the class SelectK; review it, and examine the insertion of the previously developed code for SelectK.

In \_\_init\_\_():

* init\_model\_parameters(self, data, grapharea\_size, grapharea\_matrix,…
* self.E = Parameter(embeddings\_matrix.clone().detach(), requires\_grad=**True**)
* **if** include\_globalnode\_input:  
   self.X = Parameter(data.x.clone().detach(), requires\_grad=**True**)
* *Utilities* (from the RNN, unchanged. Focusing on main elements and changes)

In the forward():

s

* *# -------------------- Input --------------------  
  # Input signal n.1: the embedding of the current (global) word*t\_word\_embeddings = self.E.index\_select(dim=0, index=t\_current\_globals\_indices)
* *# Input signal n.2: the node-state of the current global word - now with graph batching***if** self.include\_globalnode\_input:  
   t\_g\_nodestates = run\_graphnet(t\_input\_lts, batch\_elems\_at\_t,t\_globals\_indices\_ls, CURRENT\_DEVICE, self)  
   currentglobals\_nodestates\_ls.append(t\_g\_nodestates)
* *# ------------------- Senses -------------------  
  # line 1: GRU for senses + linear FF-NN to logits.***if** self.predict\_senses:  
   task\_2\_out = rnn\_loop(batch\_input\_signals, model=self)  
   task2\_out = task\_2\_out.reshape(distributed\_batch\_size \* seq\_len, task\_2\_out.shape[2])  
    
   logits\_sense = self.linear2senses(task2\_out)
* *# line 2: select senses of the k most likely globals*k\_globals\_indices = logits\_global.sort(descending=**True**).indices[:, 0:self.k]
* senses\_softmax = torch.ones((distributed\_batch\_size \* seq\_len, self.last\_idx\_senses)).to(CURRENT\_DEVICE)
* senses\_softmax = 10 \*\* (-8) \* senses\_softmax *# base probability value for non-selected senses*
* sample\_k\_indices\_lls\_relative = k\_globals\_indices.tolist()
* **for** s **in** range(distributed\_batch\_size \* seq\_len):
* k\_globals\_vocab\_indices = sample\_k\_indices\_in\_vocab\_lls[s]
* k\_globals\_words > k\_globals\_lemmatized > lemmatized\_indices
* **if** sense\_neighbours\_t.shape[0] == 0: *# no senses found, even lemmatizing. Ignore current entry* senses\_softmax[s] = torch.tensor(1 / self.last\_idx\_senses).to(CURRENT\_DEVICE)  
   **continue**
* *# standard procedure: get the logits of the senses of the most likely globals,  
  # apply a softmax only over them, and then assign an epsilon probability to the other senses*
* sample\_logits\_senses = logits\_sense.index\_select(dim=0, index=self.select\_first\_indices[**s**].to(torch.int64)).squeeze()
* logits\_selected\_senses = sample\_logits\_senses.index\_select(dim=0, index=sense\_neighbours\_t)
* softmax\_selected\_senses = tfunc.softmax(input=logits\_selected\_senses, dim=0)
* **for** i **in** range(len(sense\_neighbours\_t)):  
   i\_senseneighbours\_mask[s,sense\_neighbours\_t[i]]=**True**
* quantity\_to\_subtract\_from\_selected = epsilon \* (self.last\_idx\_senses - len(sense\_neighbours\_t))  
   softmax\_selected\_senses = subtract\_probability\_mass\_from\_selected(softmax\_selected\_senses, quantity\_to\_subtract\_from\_selected)  
   senses\_softmax[s].masked\_scatter\_(mask=i\_senseneighbours\_mask[s].data.clone(), source=softmax\_selected\_senses)
* predictions\_senses = torch.log(senses\_softmax)

### Checking code & output

k\_globals\_indices = logits\_global.sort(descending=**True**).indices[:, 0:self.K]

with batch size =2, seq\_len=5 (so 10 samples), and self.K=1, we get:

tensor([[ 846], dallas\_county

[ 846],

[ 846],

[ 846],

[ 846],

[6356], desert

[6356],

[6356],

[ 846],

[ 846]])

The random initial weights of the NN determine the most likely globals.

Turned into a LLS by

sample\_k\_indices\_in\_vocab\_lls = k\_globals\_indices.tolist()

Then, a for cycle for each element in the sample:

**for** s **in** range(distributed\_batch\_size \* seq\_len):

lemmatized\_indices\_in\_X # [42052], i.e. 42052-41206=846, ok

sense\_neighbours\_t=tensor([35377]), i.e. dallas\_county.dummySense.01

sample\_logits\_senses= logits\_sense.index\_select(dim=0, index=self.select\_first\_indices[s].to(torch.int64)).squeeze()

tensor of torch.Size([41206])

logits\_selected\_senses = sample\_logits\_senses.index\_select(dim=0, index=sense\_neighbours\_t), tensor([**0.0071**], grad\_fn=<IndexSelectBackward>)

softmax\_selected\_senses = tfunc.softmax(input=logits\_selected\_senses, dim=0), tensor([**1.**], grad\_fn=<SoftmaxBackward>)

It seems ok. Let us check with a mini-experiment.

INFO : Hyperparameters: \_batchPerSeqlen20\_area32\_lr0.0001\_epochs400

|  |  |  |  |
| --- | --- | --- | --- |
| *Epoch* | *Validation == Training* | | |
|  | Globals | Senses | Multi-senses |
| 1 | 22385.71 | 54117076.14 | 100000216.32 |
| 2 | 21683.67 | 39810810.26 | 100000216.32 |
| 10 | 3425.86 | 39810797.6 | 100000216.32 |
| 100 | 22.61 | 21544393.04 | 100000216.32 |
| 200 | 11.39 | 1000001.78 | 100000216.32 |
| 300 | 15.07 | 8576976.16 | 100000216.32 |
| 350 | 3.14 | 29.46 | 84.33 |
| 375 | 2.51 | 11.96 | 9.69 |
| 400 | 2.07 'correct\_g': 60,  'top\_k\_g': 60,  'tot\_g': 60 | 8.59 'correct\_all\_s': 53,  'top\_k\_all\_s': 53,  'tot\_all\_s': 60, | 4.38  'correct\_multi\_s': 23, 'top\_k\_multi\_s': 23, 'tot\_multi\_s': 25 |

**Analysis of predictions**

|  |  |
| --- | --- |
| Label: the next global is: <unk>(from 22213)  INFO : Label: the next sense is: group.n.01(from 12610) | INFO : The top- 2 predicted globals are:  INFO : Word: <unk> ; p=75.8%  INFO : Word: in ; p=5.27%  INFO : The top- 2 predicted senses are:  INFO : Sense: <unk>.dummySense.01 ; p = 100.0% |
| Label: the next global is: said(from 1)  INFO : Label: the next sense is: state.v.01(from 26023) | INFO : The top- 2 predicted globals are:  INFO : Word: said ; p=77.17%  INFO : Word: charge ; p=5.01%  INFO : The top- 2 predicted senses are:  INFO : Sense: state.v.01 ; p = 99.72% |
| Label: the next global is: investigation(from 4)  INFO : Label: the next sense is: probe.n.01(from 21009) | INFO : The top- 2 predicted globals are:  INFO : Word: investigation ; p=19.57%  INFO : Word: further ; p=7.53%  INFO : The top- 2 predicted senses are:  INFO : Sense: probe.n.01 ; p = 99.74% |
| Label: the next global is: of(from 5)  INFO : Label: the next sense is: of.dummySense.01(from 36523) | INFO : The top- 2 predicted globals are:  INFO : Word: of ; p=73.34%  INFO : Word: produced ; p=8.47%  INFO : The top- 2 predicted senses are:  INFO : Sense: of.dummySense.01 ; p = 100.0% |

### K=1, selecting the logits from the senses’ RNN

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Input Signals** | **Hyperparameters** |
| Globals’ prediction: **Main GRU** with 3 layers (1024>1024>512), > FF-NN to logits | 1) The word embedding of the current global (d=300) | batch\_size=32  TBPTT length=35 |
|  | grapharea=32, hops=2 |
| **Senses GRU > FF-NN** as above, we apply the softmax on the logits of the senses of the most likely ***k=1*** global | learning rate=5\*10^(-5) |

Iteration time=2.42032

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Epoch* | *Training* | | | *Validation* | | |
|  | Globals | Senses | Multi-senses | Globals | Senses | Multi-senses |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 23 |  |  |  |  |  |  |
| 24 |  |  |  |  |  |  |
| 25 |  |  |  |  |  |  |

Globals’ valid PPL = 233.75. Senses’ valid PPL = not significant, as a consequence of our manipulation of the softmax.

Compared to the simple GRU, we notice that we manage to predict correctly a number >0 of the multi-senses, and we have more correct senses in general as well. The number of guesses in the top-10 decreases. This was simply by choosing among the senses of the currently predicted globals: the choice is easy, but if the global was wrong in the first place, there is no way to retrieve the correct sense.

What happens if we choose among the first 10 globals?

### K=10, selecting the logits from the senses’ RNN

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Input Signals** | **Hyperparameters** |
| Globals’ prediction: **Main GRU** with 3 layers (1024>1024>512), > FF-NN to logits | 1) The word embedding of the current global (d=300) | batch\_size=32  TBPTT length=35 |
|  | grapharea=32, hops=2 |
| **Senses GRU > FF-NN** as above, we apply the softmax on the logits of the senses of the most likely ***k=10*** globals | learning rate=5\*10^(-5) |

Iteration time=**10.9849**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Epoch* | *Training* | | | *Validation* | | |
|  | Globals | Senses | Multi-senses | Globals | Senses | Multi-senses |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |
| 21 |  |  |  |  |  |  |
| 22 |  |  |  |  |  |  |

What happens if we use the freezing mechanism? We optimize the Senses’ GRU in the second phase, when the globals’ GRU and the word embeddings have been fixed after reaching the lowest Globals ‘Valid-PPL possible

### K=1 – with freezing – selecting the logits from the senses’ RNN

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Input Signals** | **Hyperparameters** |
| Globals: **Main GRU** with 3 layers (1024>1024>512) | 1) The word embedding of the current global (d=300) | batch\_size=32  TBPTT length=35 |
| **Senses GRU > FF-NN** as above, we apply the softmax on the logits of the senses of the most likely ***k=1*** global | grapharea=32, hops=1 |
| **Freezing:** in 1st part, update only globals. Then freeze global’s network + embeddings, and update senses | learning rate=5\*10^(-5) |
| dropout=none |

INFO : Number of trainable parameters=65.13M ;

Iteration time=1.04719, then Iteration time=2.43367

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Epoch* | *Training* | | | *Validation* | | |
|  | Globals | Senses | Multi-senses | Globals | Senses | Multi-senses |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 18 |  |  |  |  |  |  |
| 19 |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |
| 21 |  |  |  |  |  |  |
| 22 |  |  |  |  |  |  |
| 23 |  |  |  |  |  |  |
| 24 |  |  |  |  |  |  |
| 25 |  |  |  |  |  |  |
| 26 |  |  |  |  |  |  |

### K=10 – with freezing – selecting the logits from the senses’ RNN

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Input Signals** | **Hyperparameters** |
| Globals: **Main GRU** with 3 layers (1024>1024>512) | 1) The word embedding of the current global (d=300) | batch\_size=32  TBPTT length=35 |
| **Senses GRU > FF-NN** as above, we apply the softmax on the logits of the senses of the most likely ***k=10*** globals | grapharea=32, hops=1 |
| **Freezing:** in 1st part, update only globals. Then freeze global’s network + embeddings, and update senses | learning rate=5\*10^(-5) |
| dropout=none |

INFO : Number of trainable parameters=65.13M ;

Iteration time=0.80349, then Iteration time=10.73153

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Epoch* | *Training* | | | *Validation* | | |
|  | Globals | Senses | Multi-senses | Globals | Senses | Multi-senses |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| 18 |  |  |  |  |  |  |
| 19 |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |
| 21 |  |  |  |  |  |  |
| 22 |  |  |  |  |  |  |
| 23 |  |  |  |  |  |  |
| 24 |  |  |  |  |  |  |
| 25 |  |  |  |  |  |  |