[Re-structuring 3](#_Toc54029778)

[Step by step 3](#_Toc54029779)

[CreateGraphInput.py 3](#_Toc54029780)

[senselabeled\_or\_text = True 3](#_Toc54029781)

[Vocabulary 3](#_Toc54029782)

[Compute Embeddings 5](#_Toc54029783)

[WordNetData and Post-processing 5](#_Toc54029784)

[Synonyms 7](#_Toc54029785)

[Graph matrix 8](#_Toc54029786)

[Graph edges 9](#_Toc54029787)

[T.setup\_train & co 10](#_Toc54029788)

[Graph: reflections & checks 11](#_Toc54029789)

[Statistics of version 1.0 11](#_Toc54029790)

[Lemmatized forms, self-loops vs edges 11](#_Toc54029791)

[Statistics, version 1.1 12](#_Toc54029792)

[Graph test 12](#_Toc54029793)

[Reviewing the dummySenses and lemmas 15](#_Toc54029794)

[Graph review – lemmas, dummySenses and tests 18](#_Toc54029795)

[On SemCor, run 3 (after re-executing the pipeline) 18](#_Toc54029796)

[On WikiText-2, run 3 20](#_Toc54029797)

[Creating the graph edges 20](#_Toc54029798)

[Graph ranges and previous run 21](#_Toc54029799)

[Using the new grapharea\_matrix – On WikiText-2, run 4 23](#_Toc54029800)

[Statistics, v.1.2 25](#_Toc54029801)

[On WikiText-2, run 5 25](#_Toc54029802)

[On SemCor, run 4: 27](#_Toc54029803)

[Tests for mini-experiments 29](#_Toc54029804)

[Modifying the DataLoader 29](#_Toc54029805)

[MiniExperiment.py 29](#_Toc54029806)

[1: Simple GRUs 31](#_Toc54029807)

[Tests 31](#_Toc54029808)

[WikiText-2, graph check 31](#_Toc54029809)

[WikiText-2, mini-experiment 32](#_Toc54029810)

[SemCor, graph check 33](#_Toc54029811)

[SemCor, mini-experiment 35](#_Toc54029812)

[Simple GRUs on WIkiText-2 37](#_Toc54029813)

[Experiment: Simple GRUs on WIkiText-2 37](#_Toc54029814)

[Simple GRUs on SemCor 37](#_Toc54029815)

[Experiment: Simple GRUs on SemCor 37](#_Toc54029816)

[Aside: The TTO conference 38](#_Toc54029817)

[2: Variants of Simple GRUs 40](#_Toc54029818)

[With freezing 40](#_Toc54029819)

[Mini-experiment with freezing, on SemCor 40](#_Toc54029820)

[Experiment: Simple GRUs with freezing on SemCor 40](#_Toc54029821)

[2a: Loading pre-trained WT-2 model 41](#_Toc54029822)

[Pretrained Simple GRUs on SemCor: necessary modifications 41](#_Toc54029823)

[Tests: using pre-trained model 42](#_Toc54029824)

[WT-2 to WT-2: 42](#_Toc54029825)

[WT-2 to SemCor 44](#_Toc54029826)

[3: Sense context average 47](#_Toc54029827)

[Plan 47](#_Toc54029828)

[Implementation 48](#_Toc54029829)

[Tests 48](#_Toc54029830)

# Re-structuring

The part that retrieves data from WordNet and stores them is OK, and does not need to be changed.

Regarding the vocabulary, reading data from a SLC corpus, et cetera there is a list of modifications that should have been implemented long ago:

* Make 2 separate folders, for “InputData\_sense-labeled” and “InputData\_standardText”
* The sense-labeled corpus must be *rewritten, lowercased.* There should be no difference in processing. (In any case, it should be always processed lowercased)
* I must write a protocol for experiments, that includes: checking what the Neural Network reads, checking that we can overfit on a mini-fragment
* New guideline: *no more manual switches*. We should have a folder with mini-corpuses for mini-experiments (both normal text and sense-labeled). i.e. testing

## Step by step

### CreateGraphInput.py

Functions:

1. reset()
2. reset\_embeddings()
3. exe\_from\_input\_to\_vectors(do\_reset, compute\_single\_prototype, sp\_method, vocabulary\_from\_senselabeled)

We must change the vocabulary\_from\_senselabeled parameter into a senselabeled\_or\_text parameter, that decides which folder we use, etc.

Moreover, we must have reset(senselabeled\_or\_text) and reset\_embeddings(senselabeled\_or\_text)

We also change the location of the graph and grapharea\_matrix: into Graph/DatasetTypeSubfolder

### senselabeled\_or\_text = True

**SLC.organize\_splits()**

Changed into:

SLC.organize\_splits(xml\_fnames)

with xml\_fnames = [**'semcor.xml'**]

The existing code works as intended, no need to modify. We recall that:

train\_root, valid\_root, test\_root = organize\_subcorpus(xml\_fpath, 0.8)  
write\_splits\_subcorpus(xml\_fpath, train\_root, valid\_root, test\_root)

### Vocabulary

It is now opportune for us to change:

vocabulary = **V.get\_vocabulary\_df**(senselabeled\_or\_text=vocabulary\_from\_senselabeled,  
 corpus\_txt\_fpaths=[vocab\_text\_fpath],  
 out\_vocabulary\_h5\_filepath=outvocab\_filepath, min\_count=2, lowercase=**True**)

**if** os.path.exists(out\_vocabulary\_h5\_filepath):  
 vocab\_df = pd.read\_hdf(out\_vocabulary\_h5\_filepath, mode=**'r'**)  
 logging.info(**"\*\*\* The vocabulary was loaded from the file "** + out\_vocabulary\_h5\_filepath)

We should not get the vocabulary\_of\_globals.h5 file only in 1 place, but in Vocabulary/SenseLabeled or Vocabulary/StandardText

**if** senselabeled\_or\_text:  
 vocabulary\_wordfreq\_dict = build\_vocabulary\_dict\_from\_senselabeled()  
**else**:  
 vocabulary\_wordfreq\_dict = build\_vocabulary\_dict\_fromtext(corpus\_txt\_fpaths)

**build\_vocabulary\_dict\_from\_senselabeled(lowercase)**:

vocab\_dict = {}  
  
tokens\_toexclude = [Utils.EOS\_TOKEN] *# + list(string.punctuation)  
# Commas and punctuation signs are present in the Sense-Labeled Corpus as separate tokens.  
# Therefore, it makes sense to keep them in the vocabulary, and thus in the graph, as globals*slc\_split\_names = [Utils.TRAINING, Utils.VALIDATION] *# , , Utils.TEST*

Consideration: since SemCor is smaller, it may make sense to create the vocabulary not only from the training set, but also from the validation set.

Not an extremely heavy choice anyway, given the 80-10-10 split. Let’s just keep T+V

**for** token\_dict **in** SLC.read\_split(slc\_split\_name):  
 token = VocabUtils.process\_word\_token(token\_dict, lowercase)  
 **if** token **not in** tokens\_toexclude:  
 **try**:  
 prev\_freq = vocab\_dict[token]  
 vocab\_dict[token] = prev\_freq + 1

Creation of the vocabulary with process\_word\_token(td, lowercase) has worked correctly so far.

However, this is the insertion point for the *lemmatized* forms of words in the vocabulary. This was brought about by, for instance, having ‘eyelids’ but not ‘eyelid’ in SemCor, where the second form actually has WordNet senses & data.

How to handle this?

Since the frequency of a word is not relevant in the vocabulary apart from whether it is < min\_count or not, I can just add +1 to the frequency of the lemmatized word as well.

We have to take the lemmatizer for this, from:

lemmatizer = nltk.stem.WordNetLemmatizer()  
lemmatized\_forms = [LN.lemmatize\_term(word, lemmatizer) **for** word **in** vocab\_wordls]

**build\_vocabulary\_dict\_fromtext(corpus\_txt\_fpaths)**:

It doesn’t need to be changed, since it gives us 33,278 – correctly – on WikiText-2.

### Compute Embeddings

reset\_embeddings(senselabeled\_or\_text). #ok

single\_prototypes\_file #should go into the appropriate subfolder

**compute\_single\_prototype\_embeddings**(vocabulary\_df, spvs\_out\_fpath, method):

**if** method == Method.DISTILBERT:

… # currently not implemented

**else**: *# i.e. elif method == Method\_for\_SPV.FASTTEXT:*

fasttext\_vectors = EFT.load\_fasttext\_vectors()

**for** idx\_word\_freq\_tpl **in** vocabulary\_df.itertuples():

word = idx\_word\_freq\_tpl[1]

word\_vector = fasttext\_vectors[word]

word\_vectors\_lls.append(word\_vector)

embds\_nparray = np.array(word\_vectors\_lls)  
 np.save(spvs\_out\_fpath, embds\_nparray)

### WordNetData and Post-processing

vocabulary\_ls = RID.retrieve\_data\_WordNet(vocabulary)

we must adapt to the subfolders in InputData and Vocabulary

**PI.prepare**(vocabulary\_ls, sp\_method)

*# Phase 1 - Preprocessing: eliminating quasi-duplicate definitions and examples, and lemmatizing synonyms & antonyms*preprocess(vocabulary\_ls, inputdata\_folder)

*# Phase 2 - Create the Vocabulary table with the correspondences (wordSense, integer index).*create\_senses\_indices\_table(inputdata\_folder, vocabulary\_folder):

* Possible insertion point: until now, we add the dummySense if the word is one of those without senses. However, it is opportune to add it only if the *lemmatized form* has no senses. (remember to exclude words with <3chars, like ‘its’)
* It is also necessary to send the senselabeled\_or\_text specification, so we know which filepaths to use (where to find the vocabulary, etc.)

vocab\_fpath, vocabulary\_df, vocabulary\_words\_ls

and also input\_folder\_fpath.

defs\_input\_filepath, examples\_input\_filepath, defs\_input\_db, examples\_input\_db.

output\_filepath.

out\_indicesTable\_db = sqlite3.connect(output\_filepath)

out\_indicesTable\_db\_c = out\_indicesTable\_db.cursor()

out\_indicesTable\_db\_c.execute(**'''CREATE TABLE IF NOT EXISTS …**

**word\_senses\_series\_from\_defs** =   
 defs\_input\_db[Utils.DEFINITIONS][Utils.SENSE\_WN\_ID]

**word\_senses\_ls** = [sense\_str **for** sense\_str **in** word\_senses\_series\_from\_defs **if** Utils.get\_word\_from\_sense(sense\_str) **in** vocabulary\_words\_ls]

**for** wn\_id **in** word\_senses\_ls:

…out\_indicesTable\_db\_c.execute(

**"INSERT INTO indices\_table VALUES (?,?,?,?,?,?)"**,

(wn\_id, my\_vocabulary\_index,start\_defs\_count,   
 end\_defs\_count, start\_examples\_count, end\_examples\_count))

*# add the word to the set of words that do have a sense*words\_with\_senses\_set.add(Utils.get\_word\_from\_sense(wn\_id))

words\_without\_senses\_set = set(vocabulary\_words\_ls).difference(words\_with\_senses\_set)

\*\*\* XYZ

Insertion point:

for all words in words\_without\_senses\_set:

if their lemmatized form (that we use for senses-processing) has a sense   
 and they are >3 characters long,   
they do not need the addition of a dummySense.

**for** word **in** words\_without\_senses\_set:  
 *# no definitions nor examples to add here. We will add the global vector as the vector of the dummy-sense.* dummy\_wn\_id = word + **'.'** + **'dummySense'** + **'.01'**  out\_indicesTable\_db\_c.execute(**"INSERT INTO indices\_table VALUES (?,?,?,?,?,?)"**, (dummy\_wn\_id, my\_vocabulary\_index,…

Then,

*# Phase 3 - get the sentence embeddings for definitions and examples, using BERT or FasText, and store them*CE.compute\_elements\_embeddings(Utils.DEFINITIONS, embeddings\_method, inputdata\_folder)  
CE.compute\_elements\_embeddings(Utils.EXAMPLES, embeddings\_method, inputdata\_folder)  
  
*# Phase 4 - use PCA dimensionality reduction for definitions and examples, for future graph processing*apply\_PCA\_to\_defs\_examples(embeddings\_method, inputdata\_folder)

Checking the pipeline up to this point:

SemCor: |V|=24688 from the training + validation sets. ‘eyelid’ and ‘eyelids’ can both be found.   
|Definitions|=33574: say.n.01, …, say.v.11, friday.n.01, investigation.n.02, atlanta.n.02, etc.

|Examples|=29594.

note: I observe that the synonyms are… not lowercased. This must be rectified.

### Synonyms

Either the following is wrong, or I extract the word from the sense at graph-creation time:

|  |  |  |
| --- | --- | --- |
| index | sense\_wn\_id | synonyms |
| 0 | friday.n.01 | Friday |
| 1 | friday.n.01 | Fri |
| 0 | investigation.n.02 | investigating |
| 0 | atlanta.n.01 | Atlanta |
| 1 | atlanta.n.01 | capital\_of\_Georgia |
| 2 | atlanta.n.02 | Atlanta |
| 3 | atlanta.n.02 | battle\_of\_Atlanta |
| 0 | produce.n.01 | green\_goods |
| 1 | produce.n.01 | green\_groceries |
| 0 | friday.n.01 | Friday |
| 1 | friday.n.01 | Fri |

How do we process synonyms in the graph?

syn\_edges = get\_edges\_nyms(Utils.SYNONYMS, globals\_vocabulary\_df, num\_senses, inputdata\_folder), i.e.:

**for** tpl **in** nyms\_df.itertuples():  
 word\_sense = tpl.sense\_wn\_id  
 word1 = Utils.get\_word\_from\_sense(word\_sense)

word1 = VocabUtils.process\_word\_token({**'surface\_form'**: word1}, lowercasing=**True**) # we must decide how to handle this

word2 = getattr(tpl, nyms\_name)  
 word2 = VocabUtils.process\_word\_token({**'surface\_form'**: word2}, lowercasing=**True**)

and then we get their indices, and establish a bidirectional edge

edges\_ls.append((global\_idx\_1, global\_idx\_2))  
edges\_ls.append((global\_idx\_2, global\_idx\_1))

Recap: the association from WordNet, sense\_wn\_id🡪synonym, is actually viable. We get the word from the sense, and connect the 2 globals.

The problem connected to this is: how to handle the fact that SemCor is not lowercased, but only processed/received as lowercased?

We can still only process/receive it as lowercased, but this will not be determined by a (fragile) manual switch anymore, but from whether we are on SLC or not, for instance whether “SenseLabeled” is in the path of the inputdata\_folder.

WikiText-2: |V|=33,278, as expected.

|Definitions|=44756. no.1.01, …, japanese.n.01, …, flea.n.01

|Examples|=40188

### Graph matrix

**create\_graph(method, slc\_corpus):**

single\_prototypes\_file, globals\_vocabulary\_fpath, globals\_vocabulary\_df, globals\_vocabulary\_ls.

E\_embeddings = torch.tensor(np.load(os.path.join(inputdata\_folder, single\_prototypes\_file))).to(torch.float32)

use\_PCA = Utils.GRAPH\_EMBEDDINGS\_DIM < embeddings\_size

X\_definitions = load\_senses\_elements(Utils.DEFINITIONS, method, use\_PCA, inputdata\_folder)  
X\_examples = load\_senses\_elements(Utils.EXAMPLES, method, use\_PCA, inputdata\_folder)

X\_globals = torch.tensor(initialize\_globals(X\_definitions, E\_embeddings, globals\_vocabulary\_ls)).to(torch.float32)

Currently, a global is initialized as the average of all the definitions of the senses of the word.

Alternative choices are possible, such as the (possibly PCA-reduced) FastText embedding. However, when we are concatenating FastText embedding ++ global node-state as input signals of the RNN, I would prefer if they were not identical at the start.

**if** sense\_start\_defs != sense\_end\_defs:  
 sense\_def\_vectors =   
 X\_definitions[sense\_start\_defs:sense\_end\_defs]  
 **if** words\_definitionvectors\_dict[word] **is None**:  
 words\_definitionvectors\_dict[word] = sense\_def\_vectors  
 **else**:  
 words\_definitionvectors\_dict[word] =   
 np.concatenate([words\_definitionvectors\_dict[word],   
 sense\_def\_vectors])

**else**: *# if there are no definitions, use the (PCA-reduced) version of the word embedding*

**if** words\_definitionvectors\_dict[word] **is None**:  
 words\_definitionvectors\_dict[word] =   
 np.expand\_dims(E\_reduced\_embeddings[word\_vocabulary\_index,:], axis=0)

**for** word **in** words\_definitionvectors\_dict.keys():  
 average\_of\_definitions =   
 np.average(words\_definitionvectors\_dict[word], axis=0)  
 X\_globals\_ls.append(average\_of\_definitions)

X\_senses, num\_dummysenses = initialize\_senses(X\_definitions, X\_examples, X\_globals, globals\_vocabulary\_ls, average\_or\_random\_flag=**True**)

# senses are initialized with the average of the definitions, dummySenses with the globals from X\_globals

X = torch.cat([X\_senses, X\_globals, X\_definitions, X\_examples])

### Graph edges

def\_edges\_se = get\_edges\_elements(Utils.DEFINITIONS, num\_senses + num\_globals)

exs\_edges\_se = get\_edges\_elements(Utils.EXAMPLES, num\_senses + num\_globals+ X\_definitions.shape[0])

**if** slc\_corpus:  
 sc\_external\_edges =   
 get\_additional\_edges\_sensechildren\_from\_slc(globals\_vocabulary\_df,  
 globals\_start\_index\_toadd=num\_senses,   
 inputdata\_folder=inputdata\_folder)  
 sc\_edges.extend(sc\_external\_edges)

We read the training set alone, to create the connections between globals and the senses of other words (e.g. ‘atlanta’ -> location.n.01)

i.e.:

**while True**:  
 token\_dict = slc\_train\_corpus\_gen.\_\_next\_\_()

**if 'wn30\_key' in** keys:  
 wordnet\_sense = try\_to\_get\_wordnet\_sense(wn30\_key)

**else**:  
 **continue** *# there was no sense-key specified for this token*

*# 2) Get the global word of this token*word = VocabUtils.process\_word\_token(token\_dict, lowercasing=on\_senselabeled)

lemmatized\_word = lemmatize\_term(word, lemmatizer)*# since currently we always lemmatize in SelectK and other sense architectures***if** lemmatized\_word **not in** wordnet\_sense: *# we are connecting all the "external" senses, e.g. say->state.v.01*

*# We add self-loops to all globals without a sense.*edges\_selfloops = get\_edges\_selfloops(sc\_edges, num\_globals=num\_globals, num\_dummysenses=num\_dummysenses)

*# The dummy senses should already be connected to their globals*

*# And therefore, this should yield no new edges*

syn\_edges = get\_edges\_nyms(Utils.SYNONYMS, globals\_vocabulary\_df, num\_senses, inputdata\_folder)

ant\_edges = get\_edges\_nyms(Utils.ANTONYMS, globals\_vocabulary\_df, num\_senses, inputdata\_folder)

etc.etc.

**Issue**: debugging the graph

INFO : word=the

INFO : words\_definitionvectors\_dict[word]=(1, 300)

INFO : word=said

For some strange reason, I don’t find “said” in the vocabulary, only “sage, sail”

but ‘said’ is in the h5 vocabulary of senselabeled… how are we iterating?

**while** (**True**):  
 db\_row = indicesTable\_db\_c.fetchone()  
 **if** db\_row **is None**:  
 **break** word\_sense = db\_row[0]

we are iterating over the senses of the indicesTable. However, this time ‘said.dummySense.01’ is not there anymore, because ‘say’ has senses and is the lemmatized form of said’. Consequence of the latest modification.

At this point, I prefer to have all the globals initialized with the (possibly PCA-reduced) FastText embedding of the corresponding word.

### T.setup\_train & co

graph\_dataobj = DG.get\_graph\_dataobject(new=**False**, method=method, slc\_corpus=slc\_or\_text\_corpus).to(DEVICE)

grapharea\_matrix = AD.get\_grapharea\_matrix(graph\_dataobj, grapharea\_size, hops\_in\_area=1, graph\_folder=graph\_folder)

embeddings\_matrix = torch.tensor(np.load(os.path.join(inputdata\_folder, single\_prototypes\_file))).to(torch.float32)

vocabulary\_df = pd.read\_hdf(globals\_vocabulary\_fpath, mode=**'r'**)

**if** slc\_or\_text\_corpus:  
 vocabulary\_numSensesList = vocabulary\_df[**'num\_senses'**].to\_list().copy()  
 **if** all([num\_senses == -1 **for** num\_senses **in** vocabulary\_numSensesList]):  
 vocabulary\_df = AD.compute\_globals\_numsenses(graph\_dataobj, grapharea\_matrix, grapharea\_size)

This is necessary to distinguish the words that have multiple senses, so we can record separate statistics.

model = RNNs.RNN(graph\_dataobj, grapharea\_size, grapharea\_matrix, vocabulary\_df,  
 embeddings\_matrix, include\_globalnode\_input,  
 batch\_size=batch\_size, n\_layers=3, n\_hid\_units=1024)

**if** n\_gpu > 1 **and** allow\_dataparallel:  
 logging.info(**"Using "** + str(n\_gpu) + **" GPUs"**)  
 model = torch.nn.DataParallel(model, dim=0)  
 model\_forDataLoading = model.module

model.to(DEVICE)

…train\_dataloader, valid\_dataloader

T.run\_train(model, learning\_rate, train\_dataloader, valid\_dataloader, num\_epochs, predict\_senses, with\_freezing)

does not expect modifications.

However, I should pass a slc\_or\_text / vocabulary\_folder / whatever indication to the ExplorePredictions. get\_globalword\_fromindex\_df() and get\_sense\_fromindex() calls.

## Graph: reflections & checks

In the current version all the globals are initialized as the (possibly PCA-reduced) FastText embedding of the word.

### Statistics of version 1.0

On **SemCor**:

INFO : X\_senses.shape=torch.Size([**40095**, 300])

INFO : X\_globals.shape=torch.Size([**24689**, 300])

INFO : X\_definitions.shape=torch.Size([33568, 300])

INFO : X\_examples.shape=torch.Size([29585, 300])

graph ranges = [0, senses, 40095, globals, 64784, definitions, 98352, examples, 127937]

previous graph ranges = [0, senses, 41206, globals, 63988, defs, 94434, examples, 122355]

previously having

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

INFO : X\_globals.shape=torch.Size([**22782**, 200]) (they were only from the training set)

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

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

globals\_needing\_selfloop=[40096, 40106, 40113, 40125, …, 64757, 64766, 64773, 64777]

On **WikiText-2**:

INFO : X\_senses.shape=torch.Size([**46710**, 300])

INFO : X\_globals.shape=torch.Size([**33278**, 300])

INFO : X\_definitions.shape=torch.Size([28141, 300])

INFO : X\_examples.shape=torch.Size([26593, 300])

previously:

INFO : X\_senses.shape=torch.Size([28141, 300]) (it was without dummySenses)

INFO : X\_globals.shape=torch.Size([**33278**, 300])

INFO : X\_definitions.shape=torch.Size([28141, 300])

INFO : X\_examples.shape=torch.Size([26593, 300])

INFO : globals\_needing\_selfloop=[46731, 46736, 46765, 46767, … s, 79940, 79941, 79942]

### Lemmatized forms, self-loops vs edges

Why do many globals need selfloop?

Because now we are not adding a dummySense anymore for all the globals where the lemmatized form has a sense.

Those globals are left in the graph, they have their own embedding but they are isolated form the other nodes. Why?

For prediction reasons: when we predict the globals, we want to choose among all of them.

However, they should not be disconnected from their lemmatized form.

Bonus: the lemmatized form has senses, that are connected to definitions et cetera.

Wouldn’t it be better if we connected the inflected forms through a new edge type, “lemmatized”, that leads from said --to--> say ?

### Statistics, version 1.1

On **SemCor**:

INFO : X\_senses.shape=torch.Size([**40095**, 300])

INFO : X\_globals.shape=torch.Size([**24689**, 300])

INFO : X\_definitions.shape=torch.Size([33568, 300])

INFO : X\_examples.shape=torch.Size([29585, 300])

Graph ranges = 0, senses; 40095, globals; 64784, definitions; 98352, examples; 127937

On **WikiText-2**:

INFO : X\_senses.shape=torch.Size([**46710**, 300])

INFO : X\_globals.shape=torch.Size([**33278**, 300])

INFO : X\_definitions.shape=torch.Size([28141, 300])

INFO : X\_examples.shape=torch.Size([26593, 300])

Graph ranges = 0, senses;46709, globals; 79988, definitions; 108129, examples; 134722…

INFO : definition\_edges.\_\_len\_\_()=28141

INFO : example\_edges.\_\_len\_\_()=26593

INFO : sc\_edges.\_\_len\_\_()=46710

INFO : globals\_needing\_selfloop=[]

### Graph test

We write separately the code to check the graph connections, to be able to verify the correctness of the graph quickly and often instead of manually and sporadically.

**On SemCor, run 1**:

|  |
| --- |
| Center:  INFO : n=12409 ; sense=**for\_example.r.01**  INFO : Neighbours:  INFO : n=12409 ; sense=for\_example.r.01  INFO : n=77193('for\_example.r.01', 'as an example')  INFO : n=110076('for\_example.r.01', 'take ribbon snakes, for example')  INFO : n=44583 ; global=for\_example  INFO : n=49321 ; global=e.g.  INFO : n=45226 ; global=for\_instance  INFO : n=59416 ; global=e.\_g.  Center:  INFO : n=15720 ; sense=induce.v.02  INFO : Neighbours:  INFO : n=15720 ; sense=induce.v.02  INFO : n=80504('induce.v.02', 'cause to do; cause to act in a specified manner')  INFO : n=112987('induce.v.02', 'My children finally got me to buy a computer')  INFO : n=112988('induce.v.02', 'My wife made me buy a new sofa')  INFO : n=112986('induce.v.02', 'The ads induced me to buy a VCR')  INFO : n=40500 ; global=get  INFO : n=46529 ; global=stimulate  INFO : n=48879 ; global=induce  INFO : n=50438 ; global=induced  INFO : n=41408 ; global=cause  INFO : n=40554 ; global=make  INFO : n=40126 ; global=have |
| Center:  INFO : n=56966 ; global=kindred  INFO : Neighbours:  INFO : n=56966 ; global=kindred  INFO : n=16884 ; sense=kindred.s.02  INFO : n=805 ; sense=akin.s.01  INFO : n=47039 ; global=akin |
| [definition]  Center:  INFO : n=89856('retire.v.02', 'withdraw from active participation')  INFO : Neighbours:  INFO : n=89856('retire.v.02', 'withdraw from active participation')  INFO : n=25072 ; sense=retire.v.02 |
| [example]  Center:  INFO : n=103959('come\_up.v.07', 'These names came up in the discussion')  INFO : Neighbours:  INFO : n=103959('come\_up.v.07', 'These names came up in the discussion')  INFO : n=6075 ; sense=come\_up.v.07 |

we observe that including the starting node among the neighbours is ok, because we use the neighbours to select the graph\_area, that for instance is used as input to the GAT: the center node *and* the neighbours.

**On WikiTex-2, run 1**:

|  |
| --- |
| Center:  INFO : n=44183 ; sense=Decline.dummySense.01  INFO : Neighbours:  INFO : n=44183 ; sense=Decline.dummySense.01  INFO : n=56631 ; global=Decline |
| Center:  INFO : n=14808 ; sense=life.n.13  INFO : Neighbours:  INFO : n=14808 ; sense=life.n.13  INFO : definition: n=94796('life.n.13', 'a motive for living')  INFO : example: n=122590('life.n.13', 'pottery was his life')  INFO : n=47837 ; global=life |
| INFO : n=58247 ; global=migrants  INFO : Neighbours:  INFO : n=58247 ; global=migrants  note: there should be a connection between ‘migrants’ and the lemmatized / parent form ‘migrant’. Is this due to us not picking up outgoing edges, or what? |
| Center:  INFO : definition: n=85389('con.n.01', 'an argument opposed to a proposal')  INFO : Neighbours:  INFO : definition: n=85389('con.n.01', 'an argument opposed to a proposal')  INFO : n=5401 ; sense=con.n.01 |
| Center:  INFO : example: n=110001('audition.v.01', 'She auditioned for a role on Broadway')  INFO : Neighbours:  INFO : example: n=110001('audition.v.01', 'She auditioned for a role on Broadway')  INFO : n=1670 ; sense=audition.v.01 |
| Center:  INFO : example: n=122274('late.r.01', 'notice came so tardily that we almost missed the deadline')  INFO : Neighbours:  INFO : example: n=122274('late.r.01', 'notice came so tardily that we almost missed the deadline')  INFO : n=14480 ; sense=late.r.01 |

As a consequence, I decide to make all edges not-directional (i.e. bidirectional): not only synonyms and antonyms, but also lemmas, definitions-to-senses and examples-to-senses.

**On WikiText-2, Run 2**

|  |
| --- |
| **Senses** |
| INFO : n=13672 ; sense=intake.n.02  INFO : Neighbours:  INFO : n=13672 ; sense=intake.n.02  INFO : definition: n=93660('intake.n.02', 'an opening through which fluid is admitted to a tube or container')  INFO : n=66949 ; global=intake |
| INFO : n=31358 ; sense=909.dummySense.01  INFO : Neighbours:  INFO : n=31358 ; sense=909.dummySense.01  INFO : n=62302 ; global=909 |
| **Globals** |
| Center:  INFO : n=58111 ; global=stumbled  INFO : Neighbours:  INFO : n=58111 ; global=stumbled  INFO : n=36613 ; sense=stumbled.dummySense.01  error  The vocabulary has:  11401 stumbled 6 stumble -1  so why do we miss the lemmatized connection?  Note: the presence of the dummySense is a symptom of a failure further up in the pipeline:  *# ------- Remove w from the words needing a dummySense if the lemmatized form of w has senses --------* words\_without\_senses\_ls = list(words\_without\_senses\_set) words\_needing\_dummySense\_ls = [] lemmatizer = nltk.stem.WordNetLemmatizer() **for** word **in** words\_without\_senses\_ls:  lemmatized\_form = LN.lemmatize\_term(word, lemmatizer)  **if** lemmatized\_form **in** words\_with\_senses\_set:  **continue  else**:  words\_needing\_dummySense\_ls.append(word) words\_needing\_dummySense\_set = set(words\_needing\_dummySense\_ls) |
| **Definitions** |
| INFO : definition: n=107872('wolf.v.01', 'eat hastily')  INFO : Neighbours:  INFO : definition: n=107872('wolf.v.01', 'eat hastily')  INFO : n=27884 ; sense=wolf.v.01 |
| **Examples** |
| INFO : example: n=115865('dissolve.v.09', 'The heat melted the wax')  INFO : Neighbours:  INFO : example: n=115865('dissolve.v.09', 'The heat melted the wax')  INFO : n=7858 ; sense=dissolve.v.09 |

### Reviewing the dummySenses and lemmas

Inspecting, we see that we do execute lemmatization, but it is faulty, and we would be better off just accessing the lemmatized\_form in the vocabulary. Otherwise:

INFO : nuclei does not need a dummySense, because nucleus is its lemmatized parent and has senses already. (ok)

INFO : Neal needs a dummySense

INFO : Siberia needs a dummySense

INFO : Enuff needs a dummySense

INFO : keeps does not need a dummySense, because keep is its lemmatized parent and has senses already.

INFO : stumbled needs a dummySense

INFO : sympathizers needs a dummySense

etc.

note that we only see ‘stumbles’ and ‘stumbled’ in the indices\_table. Probably because I am not adding the lemmatized form to the vocabulary in the standardText, only in the Sense-labeled corpus.

In standardText, I am not adding lemmatized forms in order to have the same vocabulary size as the original WikiText-2, 33,278. If I add them, this will increase.

It is a reasonable choice if we wish to use the graph, and it shouldn’t change things too much.

**On SemCor, run 2**

SemCor already added the lemmatized forms to the dictionary, and the insertion of the lemma edges should be correct. We expect valid neighbours for the nodes…

|  |
| --- |
| **Senses** |
| INFO : n=1181 ; sense=announce.v.01  INFO : Neighbours:  INFO : n=1181 ; sense=announce.v.01  INFO : definition: n=65965('announce.v.01', 'make known; make an announcement')  INFO : example: n=99649('announce.v.01', 'She denoted her feelings clearly')  INFO : n=49061 ; global=announces  INFO : n=40493 ; global=announce |
| INFO : n=22544 ; sense=populate.v.02  INFO : Neighbours:  INFO : n=22544 ; sense=populate.v.02  INFO : definition: n=87328('populate.v.02', 'fill with inhabitants')  INFO : example: n=118530('populate.v.02', 'populate the forest with deer and wild boar for hunting')  INFO : n=45510 ; global=populate |
| INFO : n=33881 ; sense=c.dummySense.01  INFO : Neighbours:  INFO : n=33881 ; sense=c.dummySense.01  INFO : n=43304 ; global=c |
| INFO : n=39375 ; sense=corpse.dummySense.01  INFO : Neighbours:  INFO : n=39375 ; sense=corpse.dummySense.01  INFO : n=53962 ; global=corpse |
| INFO : n=23310 ; sense=propose.v.01  INFO : Neighbours:  INFO : n=23310 ; sense=propose.v.01  INFO : definition: n=88094('propose.v.01', 'make a proposal, declare a plan for something')  INFO : example: n=119208('propose.v.01', 'the senator proposed to abolish the sales tax')  INFO : n=44557 ; global=suggests  INFO : n=58926 ; global=proposes  INFO : n=41688 ; global=advise  INFO : n=40706 ; global=propose  INFO : n=41214 ; global=suggest   * there is a direct connection between a sense and a lemmatized form, e.g. propose.v.01->proposes,suggests or announce.v.01->announces. |
| **Globals** |
| INFO : n=51691 ; global=impede  INFO : Neighbours:  INFO : n=51691 ; global=impede  INFO : n=15358 ; sense=impede.v.01  INFO : n=56697 ; global=hinder  INFO : n=56481 ; global=clogging  INFO : n=47780 ; global=obstruct |
| INFO : n=56151 ; global=getting\_to  INFO : Neighbours:  INFO : n=56151 ; global=getting\_to  INFO : n=24127 ; sense=reach.v.07  INFO : n=38249 ; sense=getting\_to.dummySense.01  INFO : n=13287 ; sense=get\_to.v.02 |
| INFO : n=42191 ; global=blind  INFO : Neighbours:  INFO : n=42191 ; global=blind  INFO : n=3110 ; sense=blind.v.02  INFO : n=3105 ; sense=blind.n.02  INFO : n=3104 ; sense=blind.n.01  INFO : n=3109 ; sense=blind.v.01  INFO : n=3108 ; sense=blind.s.03  INFO : n=3107 ; sense=blind.s.02  INFO : n=3111 ; sense=blind.v.03  INFO : n=3106 ; sense=blind.n.03  INFO : n=3103 ; sense=blind.a.01  INFO : n=53900 ; global=dim  INFO : n=48435 ; global=screen  INFO : n=43313 ; global=sight  INFO : n=52504 ; global=sighted |
|  |
| **Definitions** |
| INFO : definition: n=80627('influence.n.05', 'one having power to influence another')  INFO : Neighbours:  INFO : definition: n=80627('influence.n.05', 'one having power to influence another')  INFO : n=15843 ; sense=influence.n.05 |
| INFO : definition: n=84440('musicianship.n.01', 'artistry in performing music')  INFO : Neighbours:  INFO : definition: n=84440('musicianship.n.01', 'artistry in performing music')  INFO : n=19656 ; sense=musicianship.n.01 |
| **Examples** |
| INFO : example: n=103048('channel.n.02', 'gutters carried off the rainwater into a series of channels under the street')  INFO : Neighbours:  INFO : example: n=103048('channel.n.02', 'gutters carried off the rainwater into a series of channels under the street')  INFO : n=4992 ; sense=channel.n.02 |
| INFO : example: n=107954('element.n.04', 'water is the element of fishes')  INFO : Neighbours:  INFO : example: n=107954('element.n.04', 'water is the element of fishes')  INFO : n=10314 ; sense=element.n.04 |

The graph is correct as expected. The only potential point of contention is the direct connection between sense and lemmatized form. Why does it come into existance?

Now we have:

center: n=44557 ; global=suggests

n=23310 ; sense=propose.v.01

n=35287 ; sense=suggests.dummySense.01

n=14651 ; sense=hint.v.01

n=15673 ; sense=indicate.v.05

n=29515 ; sense=suggest.v.05

n=29514 ; sense=suggest.v.03

Which begets a modification:

the dummySenses should be added in a later stage, only for globals that do not have sense connections. Example:

INFO : n=33881 ; sense=c.dummySense.01

INFO : n=33881 ; sense=c.dummySense.01

INFO : n=43304 ; global=c

OR I erase the sense connection, relying only and specifically on the lemmatized form.

Where does the connection come from, again?

In get\_edges\_sensechildren,

sourceglobal\_raw\_idx = globals\_voc\_df.loc[globals\_voc\_df[**'word'**] == word].index[0]  
sourceglobal\_idx = globals\_start\_index\_toadd + sourceglobal\_raw\_idx  
targetsense\_idx = db\_row[1]

The cause is… not here, it must be the external connection from the Sense-Labeled Corpus.

# Graph review – lemmas, dummySenses and tests

### On SemCor, run 3 (after re-executing the pipeline)

|  |
| --- |
| Senses |
| INFO : n=23202 ; sense=project.v.05  INFO : Neighbours:  INFO : n=23202 ; sense=project.v.05  INFO : definition: n=87986('project.v.05', 'cause to be heard')  INFO : example: n=119125('project.v.05', 'His voice projects well')  INFO : n=41447 ; global=project |
| INFO : n=24299 ; sense=recipe.n.01  INFO : Neighbours:  INFO : n=24299 ; sense=recipe.n.01  INFO : definition: n=89083('recipe.n.01', 'directions for making something')  INFO : n=42614 ; global=formula  INFO : n=62442 ; global=recipe |
| INFO : n=15491 ; sense=in\_principle.r.01  INFO : Neighbours:  INFO : n=15491 ; sense=in\_principle.r.01  INFO : definition: n=80275('in\_principle.r.01', 'with regard to fundamentals although not concerning details')  INFO : example: n=112710('in\_principle.r.01', 'in principle, we agree')  INFO : n=49540 ; global=in\_principle |
| Globals |
| INFO : n=55945 ; global=coming\_down  INFO : Neighbours:  INFO : n=55945 ; global=coming\_down  INFO : n=33942 ; sense=lever-action.dummySense.01  INFO : n=8564 ; sense=descend.v.01 |
| INFO : n=50125 ; global=containing  INFO : Neighbours:  INFO : n=50125 ; global=containing |
| INFO : n=60629 ; global=glendora  INFO : Neighbours:  INFO : n=60629 ; global=glendora  INFO : n=21732 ; sense=person.n.01  INFO : n=37941 ; sense=allot.dummySense.01 |
| INFO : n=43028 ; global=thereby  INFO : Neighbours:  INFO : n=43028 ; global=thereby  INFO : n=30467 ; sense=thereby.r.01 |
| Definitions |
| INFO : definition: n=84509('name.v.01', 'assign a specified (usually proper) proper name to')  INFO : Neighbours:  INFO : definition: n=84509('name.v.01', 'assign a specified (usually proper) proper name to')  INFO : n=19725 ; sense=name.v.01 |
| INFO : definition: n=96466('unclear.a.02', 'not clear to the mind')  INFO : Neighbours:  INFO : definition: n=96466('unclear.a.02', 'not clear to the mind')  INFO : n=31682 ; sense=unclear.a.02 |
| Examples |
| INFO : example: n=106756('discipline.n.02', 'for such a plan to work requires discipline')  INFO : Neighbours:  INFO : example: n=106756('discipline.n.02', 'for such a plan to work requires discipline')  INFO : n=9070 ; sense=discipline.n.02 |
| INFO : example: n=117201('ordinary.a.01', 'an ordinary wine')  INFO : Neighbours:  INFO : example: n=117201('ordinary.a.01', 'an ordinary wine')  INFO : n=20754 ; sense=ordinary.a.01 |

While definitions and examples are ok, globals are currently wrong. Must re-check: steps of edge creation.

It is simpler to operate on the WikiText-2 graph, that does not require us to read the corpus to get external connections. To this end,

### On WikiText-2, run 3

|  |
| --- |
| Senses |
| **INFO : n=18813 ; sense=oxide.n.01**  INFO : Neighbours:  INFO : n=18813 ; sense=oxide.n.01  INFO : definition: n=98801('manipulate.v.05', "control (others or oneself) or influence skillfully, usually to one's advantage")  INFO : example: n=126247('integral.s.02', 'was able to keep the collection entire during his lifetime')  INFO : example: n=126248('integral.s.02', 'fought to keep the union intact')  INFO : n=56147 ; global=costly |
| **INFO : n=14776 ; sense=investigate.v.01**  INFO : Neighbours:  INFO : n=14776 ; sense=investigate.v.01  INFO : definition: n=94764('gram.n.02', 'Danish physician and bacteriologist who developed a method of staining bacteria to distinguish among them (1853-1938)')  INFO : n=54007 ; global=bas |
| Globals |
| INFO : n=79792 ; global=separatists  INFO : Neighbours:  INFO : n=79792 ; global=separatists  INFO : n=28890 ; sense=vertically.r.01 |
| INFO : n=56671 ; global=oath  INFO : Neighbours:  INFO : n=56671 ; global=oath  INFO : n=27310 ; sense=thirteen.s.01  INFO : n=27309 ; sense=thirteen.n.01 |
| etc. |

## Creating the graph edges

On WikiText-2, we start from a Vocabulary and a indices\_table.

**Vocabulary:**

|V|=34376

0 <eos> 36718 <eos> -1

1 = 29570 = -1

2 Valkyria 54 Valkyria -1

3 Chronicles 47 Chronicles -1

etc. (the usual)

**indices\_table.sql**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1770s.n.01 | 0 | 0 | 1 | 0 | 0 |
| 1820s.n.01 | 1 | 1 | 2 | 0 | 0 |
| 1830s.n.01 | 2 | 2 | 3 | 0 | 0 |
| 1840s.n.01 | 3 | 3 | 4 | 0 | 0 |
| 1850s.n.01 | 4 | 4 | 5 | 0 | 0 |
| 1860s.n.01 | 5 | 5 | 6 | 0 | 0 |
| 1870s.n.01 | 6 | 6 | 7 | 0 | 0 |
| 1900s.n.01 | 7 | 7 | 8 | 0 | 0 |
| aa.n.01 | 8 | 8 | 9 | 0 | 0 |
| ab.n.02 | 9 | 9 | 10 | 0 | 0 |
| ab.n.04 | 10 | 10 | 11 | 0 | 0 |
| aback.r.01 | 11 | 11 | 12 | 0 | 1 |

…

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| zone.n.04 | 29961 | 29961 | 29962 | 27692 | 27692 |
| zone.v.01 | 29962 | 29962 | 29963 | 27692 | 27692 |
| zoologist.n.01 | 29963 | 29963 | 29964 | 27692 | 27692 |
| wanders.dummySense.01 | 29964 | 29964 | 29964 | 27692 | 27692 |
| E.122.dummySense.01 | 29965 | 29964 | 29964 | 27692 | 27692 |
| casemated.dummySense.01 | 29966 | 29964 | 29964 | 27692 | 27692 |

…

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Paula.dummySense.01 | 47933 | 29964 | 29964 | 27692 | 27692 |
| comprises.dummySense.01 | 47934 | 29964 | 29964 | 27692 | 27692 |

### Graph ranges and previous run

INFO : X\_senses.shape=torch.Size([47935, 300])

INFO : X\_globals.shape=torch.Size([34377, 300])

INFO : X\_definitions.shape=torch.Size([29964, 300])

INFO : X\_examples.shape=torch.Size([27692, 300])

Graph ranges:

[0, senses ; 47935, globals ; 82312, definitions ; 112276, examples, 139968]

INFO : Defining the edges: def, exs

edges-definitions = [0, 112275] -> [0, 112275]

edges-examples = [11, 139967] -> [11, 139967]

edges-get\_edges\_sensechildren = [47935, 82311] -> [0, 47934]

**Step 1: definition-to-sense edges (and example-to-sense edges)**

definition\_edges = DGE.get\_edges\_elements(Utils.DEFINITIONS, num\_senses + num\_globals, inputdata\_folder)

i.e.:

indicesTable\_db\_c.execute(**"SELECT \* FROM indices\_table"**)

**while** (**True**):  
 db\_row = indicesTable\_db\_c.fetchone()

target\_idx = db\_row[1]

start\_sources = db\_row[2] + elements\_start\_index\_toadd  
 end\_sources = db\_row[3] + elements\_start\_index\_toadd

edges\_toadd\_counter = edges\_toadd\_counter + (end\_sources-start\_sources)

**if** edges\_toadd\_counter > 0:  
 **for** source **in** range(start\_sources, end\_sources):  
 edges\_ls.append((source, target\_idx))  
 edges\_ls.append((target\_idx, source))

We have 1 definition for each sense (or 0, in case of the dummySenses – there, we add no such edges)

we should connect the sense (just db\_row[1] , the index in the table) with the definitions, and this means that elements\_start\_index\_toadd should be num\_senses + num\_globals… as it is.

followed by

example\_edges = DGE.get\_edges\_elements(Utils.EXAMPLES, num\_senses + num\_globals + X\_definitions.shape[0], inputdata\_folder)

**Error analysis: oxide.n.01**

We got:

**INFO : n=18813 ; sense=oxide.n.01**

INFO : Neighbours:

INFO : n=18813 ; sense=oxide.n.01

INFO : definition: n=98801('manipulate.v.05', "control (others or oneself) or influence skillfully, usually to one's advantage")

INFO : example: n=126247('integral.s.02', 'was able to keep the collection entire during his lifetime')

INFO : example: n=126248('integral.s.02', 'fought to keep the union intact')

INFO : n=56147 ; global=costly

In the indices\_table.sql

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| word\_sense | vocab\_index | start\_defs | end\_defs | start\_examples | end\_examples |
| oxide.n.01 | 18813 | 18813 | 18814 | 17920 | 17920 |

98801 – 82312 = 16489 (definition for manipulate.v.05)

whereas we should be having (18813 | oxide.n.01 | any compound of oxygen with another element or a radical), corresponding to 101125 …

Maybe just because the grapharea\_matrix was the old one?

### Using the new grapharea\_matrix – On WikiText-2, run 4

|  |
| --- |
| Senses |
| INFO : n=5052 ; sense=clinic.n.03  INFO : Neighbours:  INFO : n=5052 ; sense=clinic.n.03  INFO : definition: n=87364('clinic.n.03', 'a healthcare facility for outpatient care')  INFO : n=65168 ; global=clinic |
| INFO : n=7450 ; sense=defense.n.09  INFO : Neighbours:  INFO : n=7450 ; sense=defense.n.09  INFO : definition: n=89762('defense.n.09', 'an organization of defenders that provides resistance against attack')  INFO : example: n=119442('defense.n.09', 'he joined the defense against invasion')  INFO : n=52195 ; global=defense |
| INFO : n=34389 ; sense=Forerunner.dummySense.01  INFO : Neighbours:  INFO : n=34389 ; sense=Forerunner.dummySense.01  INFO : n=78053 ; global=Forerunner |
|  |
| Globals |
| INFO : n=54760 ; global=encompasses  INFO : Neighbours:  INFO : n=54760 ; global=encompasses  INFO : n=44503 ; sense=encompasses.dummySense.01   * no lemmatization here |
| INFO : n=61755 ; global=democracy  INFO : Neighbours:  INFO : n=61755 ; global=democracy  INFO : n=7559 ; sense=democracy.n.01  INFO : n=7560 ; sense=democracy.n.02  INFO : n=63196 ; global=republic  INFO : n=76487 ; global=commonwealth |
| INFO : n=71698 ; global=labourers  INFO : Neighbours:  INFO : n=71698 ; global=labourers  INFO : n=35434 ; sense=labourers.dummySense.01  23763 labourers 3 labourer -1  the lemmatization should be here… but it isn’t |
| Definitions |
| INFO : definition: n=84612('battery.n.02', 'a device that produces electricity; may have several primary or secondary cells arranged in parallel or series')  INFO : Neighbours:  INFO : definition: n=84612('battery.n.02', 'a device that produces electricity; may have several primary or secondary cells arranged in parallel or series')  INFO : n=2300 ; sense=battery.n.02 |
| Examples |
| INFO : example: n=136662('suffer.v.01', 'He suffered the penalty')  INFO : Neighbours:  INFO : example: n=136662('suffer.v.01', 'He suffered the penalty')  INFO : n=26393 ; sense=suffer.v.01 |

Definitions and examples are now ok.

We proceed therefore with manual examples:

n=71698 (global=’labourers’) and n=50794 (global=’scored’)

We have

INFO : Adding a Lemma edge between 50728=represents and 50728=represent

…

INFO : Adding a Lemma edge between 50797=qualified and 50797=qualify

so why are we unable to spot the edge from **scored**?

graph.edge\_index…

sources.tolist().index(50794) = 190609

sources[190609]

Out[13]: tensor(50794)

targets[190609]

Out[14]: tensor(50794)

It seems we have a self-loop

Although in the graph creation we got

INFO : globals\_needing\_selfloop=[]

Error spotted:

word\_idx = globals\_vocabulary\_ls.index(word) + last\_sense\_idx  
lemmatized\_idx = globals\_vocabulary\_ls.index(word) + last\_sense\_idx

Error corrected.

**Emerging issue**:

lemmatized\_idx = globals\_vocabulary\_ls.index(lemmatized\_form) + last\_sense\_idx

ValueError: 'pa' is not in list

since we have

3681 PA 27 PA -1

4550 pas 180 pa -1

because there are 2 instances of lemmatization when creating the vocabulary, and the nltk WordNet lemmatizer is not 100% consistent between runs.

I thus modify the vocabulary, so that we add lemmatized forms to the general list of words and to the list of lemmatized forms after lemmatizing them *only once*.

On WT-2, ok.

Although there is a weird occurrence:

4549 pass 526 pas -1

4550 pas 180 pas -1

…

12776 passes 109 pass -1

*It seems we should rely on the lemmatized form only if the inflected/base form does not have senses already.*

### Statistics, v.1.2

**WikiText-2**:

INFO : X\_senses.shape=torch.Size([**47936**, 300])

INFO : X\_globals.shape=torch.Size([**34377**, 300])

INFO : X\_definitions.shape=torch.Size([29964, 300])

INFO : X\_examples.shape=torch.Size([27692, 300])

Graph ranges: [47936, 82313, 112277, 139969]

**SemCor**:

INFO : X\_senses.shape=torch.Size([**40098**, 300])

INFO : X\_globals.shape=torch.Size([**24689**, 300])

INFO : X\_definitions.shape=torch.Size([33568, 300])

INFO : X\_examples.shape=torch.Size([29585, 300])

Graph ranges: [40098, 64787, 98355, 127940]

### On WikiText-2, run 5

|  |
| --- |
| Senses |
| INFO : n=31795 ; sense=monastic.dummySense.01  INFO : Neighbours:  INFO : n=31795 ; sense=monastic.dummySense.01  INFO : n=70031 ; global=monastic   * dummySense, connected to its global, and nothing else |
| INFO : n=3636 ; sense=burden.n.04  INFO : Neighbours:  INFO : n=3636 ; sense=burden.n.04  INFO : definition: n=85949('burden.n.04', 'the central idea that is expanded in a document or discourse')  INFO : n=68199 ; global=burden   * sense, connected to its definition and its global |
| INFO : n=8340 ; sense=disqualified.s.02  INFO : Neighbours:  INFO : n=8340 ; sense=disqualified.s.02  INFO : definition: n=90653('disqualified.s.02', 'barred from competition for violation of rules')  INFO : example: n=120296('disqualified.s.02', 'a disqualified player')  INFO : n=79845 ; global=disqualified |
| INFO : n=8827 ; sense=dress.v.14  INFO : Neighbours:  INFO : n=8827 ; sense=dress.v.14  INFO : definition: n=91140('dress.v.14', 'apply a bandage or medication to')  INFO : example: n=120800('dress.v.14', "dress the victim's wounds")  INFO : n=54206 ; global=dress |
| Globals |
| INFO : n=82224 ; global=supernovae  INFO : Neighbours:  INFO : n=82224 ; global=supernovae  INFO : n=68416 ; global=supernova   * inflected form, connected to its parent/lemmatized form. |
| INFO : n=48694 ; global=improvements  INFO : Neighbours:  INFO : n=48694 ; global=improvements  INFO : n=48695 ; global=improvement   * inflected form, <----> lemmatized form |
| INFO : n=58093 ; global=threshold  INFO : Neighbours:  INFO : n=58093 ; global=threshold  INFO : n=27349 ; sense=threshold.n.02  INFO : n=27348 ; sense=threshold.n.01  INFO : n=64142 ; global=doorway  INFO : n=61906 ; global=brink   * global with senses and synonyms |
| INFO : n=62632 ; global=zinc  INFO : Neighbours:  INFO : n=62632 ; global=zinc  INFO : n=29956 ; sense=zinc.v.01  INFO : n=29955 ; sense=zinc.n.01 |
| Definitions |
| INFO : definition: n=104228('referee.v.02', "evaluate professionally a colleague's work")  INFO : Neighbours:  INFO : definition: n=104228('referee.v.02', "evaluate professionally a colleague's work")  INFO : n=21915 ; sense=referee.v.02 |
| INFO : definition: n=88897('cover.v.12', 'protect or defend (a position in a game)')  INFO : Neighbours:  INFO : definition: n=88897('cover.v.12', 'protect or defend (a position in a game)')  INFO : n=6584 ; sense=cover.v.12 |
| Examples |
| INFO : example: n=137174('technical.a.03', 'technological development')  INFO : Neighbours:  INFO : example: n=137174('technical.a.03', 'technological development')  INFO : n=26991 ; sense=technical.a.03 |
| INFO : example: n=127886('lyric.n.01', 'the song uses colloquial language')  INFO : Neighbours:  INFO : example: n=127886('lyric.n.01', 'the song uses colloquial language')  INFO : n=16312 ; sense=lyric.n.01 |

This time, everything is as expected, the graph on WikiText-2 passes muster.

### On SemCor, run 4:

INFO : Adding manually UNK\_TOKEN=<unk>

INFO : Generator over subcorpus at TextCorpuses/SenseLabeled/Validation/semcor.xml

INFO : len(vocab\_dict.keys())=49092

INFO : len(lemmatized\_forms\_ls)=49091 *# before the frequency cut. However, adding <unk> causes a +1 problem, I should add it as lemmatized form as well.*

INFO : Removing from the vocabulary words with frequency < 2 *# must also adjust this*

INFO : \*\*\* The vocabulary was created. Number of words= 24689\*\*\*

|  |
| --- |
| Senses |
| INFO : n=7015 ; sense=convertible.n.03  INFO : Neighbours:  INFO : n=7015 ; sense=convertible.n.03  INFO : definition: n=71802('convertible.n.03', 'a sofa that can be converted into a bed')  INFO : n=55885 ; global=convertible |
| INFO : n=21477 ; sense=patriot.n.01  INFO : Neighbours:  INFO : n=21477 ; sense=patriot.n.01  INFO : definition: n=86264('patriot.n.01', 'one who loves and defends his or her country')  INFO : n=54485 ; global=patriot |
| INFO : n=26700 ; sense=settle.v.21  INFO : Neighbours:  INFO : n=26700 ; sense=settle.v.21  INFO : definition: n=91487('settle.v.21', 'form a community')  INFO : example: n=122006('settle.v.21', 'The Swedes settled in Minnesota')  INFO : n=44542 ; global=settle |
| Globals |
| INFO : n=48694 ; global=prejudice  INFO : Neighbours:  INFO : n=48694 ; global=prejudice  INFO : n=22820 ; sense=prejudice.v.01  INFO : n=22821 ; sense=prejudice.v.02  INFO : n=2856 ; sense=bias.n.01  INFO : n=48424 ; global=bias  INFO : n=58237 ; global=prejudiced  INFO : n=60442 ; global=prejudices   * global with senses (including external ones from SLC), and lemmatized forms |
| INFO : n=40333 ; global=place  INFO : Neighbours:  INFO : n=40333 ; global=place  INFO : n=22093 ; sense=place.v.11  INFO : n=30107 ; sense=target.v.01  INFO : n=22092 ; sense=place.v.09  INFO : n=22086 ; sense=place.n.12  INFO : n=24065 ; sense=rate.v.01  INFO : n=22095 ; sense=place.v.15  INFO : n=17865 ; sense=locate.v.03  INFO : n=22089 ; sense=place.v.02  INFO : n=28767 ; sense=stead.n.01  INFO : n=22088 ; sense=place.n.15  INFO : n=26306 ; sense=seat.n.01  INFO : n=22082 ; sense=place.n.03  INFO : n=22603 ; sense=position.n.01  INFO : n=22091 ; sense=place.v.06  INFO : n=22085 ; sense=place.n.10  INFO : n=22084 ; sense=place.n.06  INFO : n=23623 ; sense=put.v.01  INFO : n=22087 ; sense=place.n.13  INFO : n=22081 ; sense=place.n.02  INFO : n=22606 ; sense=position.n.06  INFO : n=22094 ; sense=place.v.12  INFO : n=28091 ; sense=space.n.07  INFO : n=22083 ; sense=place.n.04  INFO : n=22096 ; sense=place.v.16  INFO : n=15229 ; sense=identify.v.01  INFO : n=14772 ; sense=home.n.01  INFO : n=22090 ; sense=place.v.05  INFO : n=40909 ; global=property  INFO : n=40995 ; global=locate  INFO : n=41251 ; global=set  INFO : n=47094 ; global=seat   * Global with a large number of senses, and some synonyms |
| INFO : n=55301 ; global=estate  INFO : Neighbours:  INFO : n=55301 ; global=estate  INFO : n=10833 ; sense=estate.n.02  INFO : n=10832 ; sense=estate.n.01  INFO : n=42978 ; global=land  INFO : n=44630 ; global=acre  INFO : n=55300 ; global=estates   * global with senses, synonyms and inflected form |
| INFO : n=55988 ; global=crashing  INFO : Neighbours:  INFO : n=55988 ; global=crashing  INFO : n=43877 ; global=crash   * global – inflected form |
| Definitions |
| INFO : definition: n=67316('be.v.05', 'happen, occur, take place')  INFO : Neighbours:  INFO : definition: n=67316('be.v.05', 'happen, occur, take place')  INFO : n=2529 ; sense=be.v.05 |
| Examples |
| INFO : example: n=123435('squarely.r.03', 'the bat met the ball squarely')  INFO : Neighbours:  INFO : example: n=123435('squarely.r.03', 'the bat met the ball squarely')  INFO : n=28490 ; sense=squarely.r.03 |

Graph on SemCor confirmed.

# Tests for mini-experiments

### Modifying the DataLoader

We should be able to load the MiniCorpuses from their separate location, without the need to change them manually in the folder.

### MiniExperiment.py

Taken from Training.py. We should exclude anything related to Validation, and instead send a flag in the forward() call that allows the models to print the output they read.

n: we must also update Utils.get\_startpoint\_dummySenses(), since we do not have only indices\_table.sql anymore.

How can we log the input & labels, to understand what we pass to the RNN from the corpus?

batch\_input.shape= torch.Size([2, 5, 1150])

The rows of the selected grapharea matrix should contain, at the first place, the current word’s node (since their structure is: node & neighbours | edge sources | edge targets | edge type).

This would mean

batch\_input[:,:,0]

and since the rows’ length is actually 575, I should take [0] and also [575], for the sense label??

Or maybe it is simpler to operate on the batch\_labels, of shape [10,2].

On WikiText-2, we have labels:

tensor([[ 152, -1],

[ 618, -1],

[ 76, -1],

[ 17, -1],

[5437, -1],

[ 85, -1],

[ 177, -1],

[3256, -1],

[ 46, -1],

[ 17, -1]])

Given the input correspondence:

batch\_input[:,:,0] - model.last\_idx\_senses =

tensor([[12088, 152, 618, 76, 17],

[ 5437, 85, 177, 3256, 46]])

How is the input created again?

…

global\_idx, sense\_idx = self.current\_token\_tpl  
relative\_global\_idx = global\_idx + self.nn\_model.last\_idx\_senses  
(global\_forwardinput\_triple, sense\_forwardinput\_triple)= \  
 get\_forwardinput\_forelement(relative\_global\_idx, sense\_idx, self.grapharea\_matrix, self.area\_size)

where we got:

area\_x\_indices\_global, edge\_index\_global, edge\_type\_global =   
 AD.get\_node\_data(grapharea\_matrix, global\_idx, area\_size)

area\_x\_indices\_sense, edge\_index\_sense, edge\_type\_sense =   
 AD.get\_node\_data(grapharea\_matrix, sense\_idx, area\_size)

and:

globals\_input\_ls.append(pack\_input\_tuple\_into\_tensor(global\_input\_tpl,   
 self.grapharea\_size))  
senses\_input\_ls.append(pack\_input\_tuple\_into\_tensor(sense\_input\_tpl,   
 self.grapharea\_size))

It depends on:

**def** pack\_input\_tuple\_into\_tensor(input\_tuple, graph\_area):  
  
 max\_edges = int(graph\_area\*\*1.5)  
 in\_tensor = - 1 \* torch.ones(size=(graph\_area + max\_edges\*3,)).to(torch.long)  
 x\_indices = input\_tuple[0]  
 edge\_sources = input\_tuple[1][0]  
 edge\_destinations = input\_tuple[1][1]  
 edge\_type = input\_tuple[2]  
 **if** len(edge\_sources) > max\_edges:  
 logging.warning(**"Num edges="** + str(len(edge\_sources)) + **" , while max\_edges packed="** + str(max\_edges))  
 in\_tensor[0:len(x\_indices)] = x\_indices  
 in\_tensor[graph\_area: graph\_area+min(len(edge\_sources), max\_edges)] = edge\_sources[0:max\_edges]  
 in\_tensor[graph\_area+max\_edges:graph\_area+max\_edges+min(len(edge\_destinations), max\_edges)] = edge\_destinations[0:max\_edges]  
 in\_tensor[graph\_area+2\*max\_edges:graph\_area+2\*max\_edges+min(len(edge\_type), max\_edges)] = edge\_type[0:max\_edges]  
 **return** in\_tensor

# 1: Simple GRUs

## Tests

### WikiText-2, graph check

Senses:

|  |
| --- |
| INFO : n=24913 ; sense=sociology.n.01  INFO : Neighbours:  INFO : n=24913 ; sense=sociology.n.01  INFO : definition: n=107226('sociology.n.01', 'the study and classification of human societies')  INFO : n=71575 ; global=sociology |
| INFO : n=30238 ; sense=2100.dummySense.01  INFO : Neighbours:  INFO : n=30238 ; sense=2100.dummySense.01  INFO : n=68216 ; global=2100 |
| INFO : n=22236 ; sense=removed.s.01  INFO : Neighbours:  INFO : n=22236 ; sense=removed.s.01  INFO : definition: n=104549('removed.s.01', 'separated in relationship by a given degree of descent')  INFO : example: n=133094('removed.s.01', 'a cousin once removed')  INFO : n=48715 ; global=removed |
| INFO : n=21835 ; sense=recovered.s.02  INFO : Neighbours:  INFO : n=21835 ; sense=recovered.s.02  INFO : definition: n=104148('recovered.s.02', 'found after being lost')  INFO : n=59128 ; global=recovered |

Globals:

|  |
| --- |
| INFO : n=82224 ; global=supernovae  INFO : Neighbours:  INFO : n=82224 ; global=supernovae  INFO : n=68416 ; global=supernova |
| INFO : n=56430 ; global=gendered  INFO : Neighbours:  INFO : n=56430 ; global=gendered  INFO : n=39819 ; sense=gendered.dummySense.01 |
| INFO : n=49412 ; global=destination  INFO : Neighbours:  INFO : n=49412 ; global=destination  INFO : n=7759 ; sense=destination.n.02  INFO : n=48638 ; global=finish  INFO : n=51991 ; global=address  INFO : n=52137 ; global=terminus  INFO : n=65244 ; global=destinations |
| INFO : n=63167 ; global=stepping  INFO : Neighbours:  INFO : n=63167 ; global=stepping  INFO : n=52108 ; global=step |

Definitions:

|  |
| --- |
| INFO : definition: n=89723('deep.s.04', 'very distant in time or space')  INFO : Neighbours:  INFO : definition: n=89723('deep.s.04', 'very distant in time or space')  INFO : n=7410 ; sense=deep.s.04 |

Examples:

|  |
| --- |
| INFO : example: n=117112('clearly.r.01', 'they were clearly lost')  INFO : Neighbours:  INFO : example: n=117112('clearly.r.01', 'they were clearly lost')  INFO : n=5000 ; sense=clearly.r.01 |

passed.

### WikiText-2, mini-experiment

**Input & labels**

Batch globals: ['Markgraf', 'was', 'present', 'during', 'the', 'fleet', 'operation', 'that', 'resulted', 'in', 'the', 'Battle', 'of', 'Jutland', 'which', 'took', 'place', 'on', '31', 'May']

INFO : Batch senses: []

…

Batch globals: ['.', 'At', '17', ':', '45', ',', 'Scheer', 'ordered', 'a', 'two', '@-@', 'point', 'turn', 'to', 'port', 'to', 'bring', 'his', 'ships', 'closer']

**Overfitting on a training fragment**

Globals only. Batch size=4, seq\_len=5, lr=0.0001

|  |  |  |  |
| --- | --- | --- | --- |
| *Epoch* | *Training PPL* | | |
|  | Globals | Senses | Multi-senses |
| 1 | 32134.02  'correct\_g': 23, 'top\_k\_g': 39, 'tot\_g': 300 | - | - |
| 10 | 107.07 |  |  |
| 100 | 94.63  'correct\_g': 29, 'top\_k\_g': 103, 'tot\_g': 300 |  |  |
| 150 | 1.71 |  |  |
| 200 | 1.08  'correct\_g': 300, 'top\_k\_g': 300, 'tot\_g': 300 |  |  |

**Analysis of predictions:**

|  |  |
| --- | --- |
| Label: the next global is: was(from 152) | INFO : The top- 10 predicted globals are:  INFO : Word: was ; p=81.55% |
| …  Label: the next global is: Jutland(from 12095) | INFO : The top- 10 predicted globals are:  INFO : Word: Jutland ; p=86.55% |
| Label: the next global is: 1(from 1145) | INFO : The top- 10 predicted globals are:  INFO : Word: 1 ; p=91.5% |
| Label: the next global is: June(from 1130) | INFO : The top- 10 predicted globals are:  INFO : Word: June ; p=93.88% |

etc., ok.

### SemCor, graph check

Senses:

|  |
| --- |
| INFO : n=22652 ; sense=post.n.11  INFO : Neighbours:  INFO : n=22652 ; sense=post.n.11  INFO : definition: n=87439('post.n.11', 'the delivery and collection of letters and packages')  INFO : example: n=118619('post.n.11', "if you hurry you'll catch the post")  INFO : example: n=118618('post.n.11', 'it came by the first post')  INFO : n=40874 ; global=post |
| INFO : n=23133 ; sense=productive.s.03  INFO : Neighbours:  INFO : n=23133 ; sense=productive.s.03  INFO : definition: n=87920('productive.s.03', 'yielding positive results')  INFO : n=42253 ; global=productive |
| INFO : n=34971 ; sense=satire.dummySense.01  INFO : Neighbours:  INFO : n=34971 ; sense=satire.dummySense.01  INFO : n=44668 ; global=satire |
| INFO : n=13716 ; sense=gravitation.n.03  INFO : Neighbours:  INFO : n=13716 ; sense=gravitation.n.03  INFO : definition: n=78503('gravitation.n.03', 'a figurative movement toward some attraction')  INFO : example: n=111157('gravitation.n.03', 'the gravitation of the middle class to the suburbs')  INFO : n=63773 ; global=gravitation |

Globals:

|  |
| --- |
| INFO : n=48694 ; global=prejudice  INFO : Neighbours:  INFO : n=48694 ; global=prejudice  INFO : n=22820 ; sense=prejudice.v.01  INFO : n=22821 ; sense=prejudice.v.02  INFO : n=2856 ; sense=bias.n.01  INFO : n=48424 ; global=bias  INFO : n=58237 ; global=prejudiced  INFO : n=60442 ; global=prejudices |
| INFO : n=44606 ; global=surplus  INFO : Neighbours:  INFO : n=44606 ; global=surplus  INFO : n=10971 ; sense=excess.n.01  INFO : n=37016 ; sense=surplus.dummySense.01  INFO : n=10973 ; sense=excess.s.01  INFO : n=45551 ; global=excess  INFO : n=51748 ; global=surpluses |
| INFO : n=56817 ; global=grammatical  INFO : Neighbours:  INFO : n=56817 ; global=grammatical  INFO : n=13634 ; sense=grammatical.a.02  INFO : n=13633 ; sense=grammatical.a.01  INFO : n=60362 ; global=grammatic |
| INFO : n=46226 ; global=kindly  INFO : Neighbours:  INFO : n=46226 ; global=kindly  INFO : n=16882 ; sense=kindly.r.01  INFO : n=16883 ; sense=kindly.s.02  INFO : n=5072 ; sense=charitable.s.03  INFO : n=52882 ; global=charitable |

Definitions:

|  |
| --- |
| INFO : definition: n=93390('stall.v.06', 'cause an airplane to go into a stall')  INFO : Neighbours:  INFO : definition: n=93390('stall.v.06', 'cause an airplane to go into a stall')  INFO : n=28603 ; sense=stall.v.06 |

Examples:

|  |
| --- |
| INFO : example: n=99302('alert.s.03', 'was now awake to the reality of his predicament')  INFO : Neighbours:  INFO : example: n=99302('alert.s.03', 'was now awake to the reality of his predicament')  INFO : n=845 ; sense=alert.s.03 |

passed.

### SemCor, mini-experiment

Error at:

get\_missing\_sense\_label(global\_absolute\_index, grapharea\_matrix, last\_sense\_idx, first\_idx\_dummySenses)

with:

global\_absolute\_index=14

global\_relative\_index=40112

adjacent\_nodes=tensor([40112, 20086, 20088, 20089, 20087, 20090, 44072, 40303, 42566, 40327])

first\_idx\_dummySenses=33569

last\_sense\_idx=40098

dummySenses=<class 'list'>: []

Unexpected [] … First, we shouldn’t even be here because we have senses that are not dummySenses. What is the word?

word=’no’

**'wn30\_key' in** keys? False

**if** slc\_or\_text:  
 sense\_index = get\_missing\_sense\_label(…)

Let us run a graph check on the node, global n.14.

INFO : n=40112 ; global=no

INFO : Neighbours:

INFO : n=40112 ; global=no # the node itself

INFO : n=20086 ; sense=no.a.01 # senses …

INFO : n=20088 ; sense=no.r.01

INFO : n=20089 ; sense=no.r.02

INFO : n=20087 ; sense=no.n.01

INFO : n=20090 ; sense=no.r.03

INFO : n=44072 ; global=no\_more # synonym

INFO : n=40303 ; global=all # antonyms

INFO : n=42566 ; global=yes

INFO : n=40327 ; global=some

The start of the SemCor corpus does not have a sense label for ‘no’ at the given location:

<word surface\_form="no" pos="DT"/>

We pick a sense as the label, and

**return** senses[0]

i.e. tensor(20086). We should return the item(). Otherwise,

(global\_index, sense\_index)= (14, tensor(20086))

But we still have a problem. Where?

global\_absolute\_index=18 ; senses=[]

INFO : n=40116 ; global=irregularities

INFO : Neighbours:

INFO : n=40116 ; global=irregularities

INFO : n=40117 ; global=irregularity

**Problem: inflected forms do not have senses attached…**

wordnet\_sense = abnormality.n.04

sense\_index\_queryresult=None

Therefore,

get\_missing\_sense\_label(global\_absolute\_index, grapharea\_matrix, last\_sense\_idx, first\_idx\_dummySenses)

however, we should select the lemmatized form from the hdf5 to avoid this problem.

Solved.

New error at:

global\_absolute\_index=41

‘was’ instead of ‘be’, with no in-corpus sense label, because I am not *always* lemmatizing.

**Input & labels**

Epoch 1:

Batch globals: ['the', '<unk>', 'said', 'friday', 'an', 'investigation']

INFO : Batch senses: ['the.dummySense.01', 'group.n.01', 'state.v.01', 'friday.n.01', 'an.dummySense.01', 'probe.n.01']

…

Batch globals: ['manner', 'in', 'which', 'the', 'election', 'was']

INFO : Batch senses: ['manner.n.01', 'actually.r.02', 'which.dummySense.01', 'the.dummySense.01', 'election.n.01', 'be.v.05']

Epoch 2:

…

Batch globals: ['manner', 'in', 'which', 'the', 'election', 'was']

INFO : Batch senses: ['manner.n.01', 'actually.r.02', 'which.dummySense.01', 'the.dummySense.01', 'election.n.01', 'be.v.05']

#note: I am skipping “conducted”, but that is to be expected since we are printing the input, not the labels.

**Overfitting on a training fragment**

Globals and senses, joint optimization. Batch size=2, seq\_len=3, lr=0.0001

|  |  |  |  |
| --- | --- | --- | --- |
| *Epoch* | *Training PPL* | | |
|  | Globals | Senses | Multi-senses |
| 1 | 24045.83 | 39522.51 | 40973.28 |
| 10 | 51.83 | 64.16 | 158.73 |
| 100 | 31.96  'correct\_g': 7, 'top\_k\_g': 31, 'tot\_g': 60 | 34.89  'correct\_all\_s': 7, 'top\_k\_all\_s': 24, 'tot\_all\_s': 60 | 52.95  'correct\_multi\_s': 0, 'top\_k: 2, 'tot\_multi\_s': 19 |
| 200 | 3.26 | 1.2 | 1.25 |
| 248 | 1.09  'correct\_g': 59, 'top\_k\_g': 60, 'tot\_g': 60 | 1.07  'correct\_all\_s': 59, 'top\_k\_all\_s': 60, 'tot\_all\_s': 60 | 1.09  'correct\_multi\_s': 18, 'top\_k\_multi\_s': 19, 'tot\_multi\_s': 19 |

## Simple GRUs on WIkiText-2

### Experiment: Simple GRUs on WIkiText-2

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Hyperparameters** | **Senses’ method** |
| For the globals: main GRU,  1024>1024>512 | batch size=32  sequence length = 35  Learning rate= 5e-05 | none, the GRU senses is unused here |
|  |  |  |

Parameters=

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Epoch* | *Training PPL* | | | *Validation PPL* | | |
|  | Globals | Senses | Multi-senses | Globals | Senses | Multi-senses |
| 1 | 1344.79 | - | - | 617.13 | - | - |
| 2 | 598.48 |  |  | 399.91 |  |  |
| 10 | 190.12 |  |  | 211.06 |  |  |
| 22 | 98.14 |  |  | 183.89 |  |  |
| 23 | 93.8 |  |  | **183.76**  'correct\_g': 47579, 'top\_k\_g': 104622, 'tot\_g': 212800 |  |  |
| 24 | 89.72 |  |  | 183.82 |  |  |
| 25 | 85.86 |  |  | 184.02 |  |  |

## Simple GRUs on SemCor

### ­Experiment: Simple GRUs on SemCor

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Hyperparameters** | **Senses’ method** |
| For the globals: main GRU,  1024>1024>512  For the senses: senses’ GRU | batch size=32  sequence length = 35  Learning rate= 5e-05 | Senses’ GRU, 1024>1024>512.  Joint optimization of the globals and senses’ task |
|  |  |  |

Parameters=

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Epoch* | *Training PPL* | | | *Validation PPL* | | |
|  | Globals | Senses | Multi-senses | Globals | Senses | Multi-senses |
| 1 | 1036.38 | 1161.71 | 10898.96 | 837.53 | 839.3 | 7729.86 |
| 2 | 663.22 | 731.52 | 7838.79 | 736.41 | **829.05**  **'correct\_all\_s': 6077,** 'top\_k\_all\_s': 28260, 'tot\_all\_s': 80640, 'correct\_multi\_s': 0, 'top\_k\_multi\_s': 1240, 'tot\_multi\_s': 25345 | 7246.18 |
| 3 | 531.43 | 722.71 | 7734.6 | 598.84 | 840.7  **'correct\_all\_s': 6077,** **'top\_k\_all\_s': 28536,** **'correct\_multi\_s': 0,** **'top\_k\_multi\_s': 1476, '** | 7285.65 |
| 4 | 441.27 | 719.87 | 7691.66 | 533.44 | 850.83 | 7429.72 |
| 17 | 174.18 | 721.82 | 7750.68 | 379.88 | 957.01 | 8140.39 |
| 18 | 167.51 | 720.9 | 7727.42 | **379.84**  'correct\_g': 16145, 'top\_k\_g': 34978, 'tot\_g': 80640 | 963.94 | 8226.89 |
| 19 | 161.26 | 718.35 | 7672.82 | 380.33 | 955.34 | 7989.96 |

The same GRU that achieves 184 Valid-PPL on WikiText-2 ends up having 379 Valid-PPL on SemCor.

We recall that SemCor has only 650K training tokens.

The next steps will be:

* try this model with freezing. If we try to achieve the best possible Valid-PPL on the standard language modeling task alone, what do we get?
* try the model that was pre-trained on WikiText-2, both with joint optimization and with freezing on SemCor. We should absolutely try to get a better performance on the globals, considering the fact that the next methods use Structured Prediction, like SelectK.
* explore the inclusion of the input signal: global node state

### Aside: The TTO conference

I) stance classification on opinions, for rumour verification

if a claim attracts a lot of scepticism, it is more likely to be false

querying, supporting, commenting, denying tweets on a claim

The RumourEval task was proposed to test the hypothesis regarding the synergy between stance and rumour veracity.

Model: Variational Autoencoder + Multi-task set-up

Dungs et al., and later Lille et al., used a Hidden Markov Model to proceed (automatically labeled stance) 🡪 rumour veracity classification

The ideal verification model should provide justifications/explanations for its predictions, and not exhibit significant biases.

Problem: tweets can have a stance towards the previous tweet (“you idiot!” etc.) or also towards the rumour.

What is the input signal we are looking for? The number of people refuting, the number of supporters, the style of refuting?

The signal would include the sequence component, i.e. the history of people mass-commenting on it.

Although information bubbles would bring scores of supporters even for absurd claims. Big data, i.e. mass numbers of comments could contain this problem.

II) Hypothesis: propaganda is subjective, emotional, unfactual.

Goal: to build a propaganda detector

Datasets: speeches (in decreasing order of propaganda: Goebbels, Trump, Churchill, Obama, news)

Methods: Logistic regression, SVM

Features: word counts, word level TF-IDF, etc.

Alternatively, a Bi-LSTM for training ad prediction, utilizing pre-training word embeddings, etc.

Sentence level detection is harder than article level.

Simply combining the datasets is not effective to improve cross-domain performance.

Linguistic analysis: most distinctive words, like “advantage, new, political, shocking, impossible, lies, devastating, autonomous, ridiculous”.

# 2: Variants of Simple GRUs

## With freezing

### Mini-experiment with freezing, on SemCor

**Overfitting on a training fragment**

Globals first, then senses (freezing mechanism). Batch size=2, seq\_len=8, lr=0.0001

|  |  |  |  |
| --- | --- | --- | --- |
| *Epoch* | *Training PPL* | | |
|  | Globals | Senses | Multi-senses |
| 1 | 24045.81 | - | - |
| 2 | 19716.24 |  |  |
| 10 | 51.83 |  |  |
| 100 | 31.96 |  |  |
| 200 | 31.72 |  |  |
| 300 | 12.63 |  |  |
| 399 | 1.2 | - | - |
| 400 | 1.2 | 39728.94 | 40994.33 |
| 401 | 1.2 | 34389.99 | 37223.62 |
| 450 | 1.2 | 34.84 | 53.13 |
| 500 | 1.2 | 1.26 | 1.32 |
| 600 | 1.2 | 1.05  'correct\_all\_s': 59, 'top\_k\_all\_s': 60, 'tot\_all\_s': 60 | 1.07  'correct\_multi\_s': 18, 'top\_k\_multi\_s': 19, 'tot\_multi\_s': 19 |

There is 1 sense we can not get. What is it?

Still correct:

INFO : Label: the next sense is: that.dummySense.01(from 35499)

INFO : The top- 10 predicted senses are:

INFO : Sense: that.dummySense.01 ; p = 48.84%

INFO : Sense: no.a.01 ; p = 48.21%

I don’t see an error… even if I am not registering 1 correct prediction in the statistics, it is not a high priority to fix.

### Experiment: Simple GRUs with freezing on SemCor

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Hyperparameters** | **Senses’ method** |
| For the globals: main GRU,  1024>1024>512  For the senses: senses’ GRU w/ freezing mechanism | batch size=32  sequence length = 35  Learning rate= 5e-05 | Senses’ GRU, 1024>1024>512. + freezing, we first optimize the standard LM on globals and then the senses |
|  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Epoch* | *Training PPL* | | | *Validation PPL* | | |
|  | Globals | Senses | Multi-senses | Globals | Senses | Multi-senses |
| 1 | 1036.38 | - | - | 837.55(837.53) | - | - |
| 2 | 663.33 | - | - | 736.26(736.41) | **-** | - |
| 17 | 174.43 (174.18) |  |  | **380.22 (**379.88) |  |  |
| 18 | 167.76 |  |  | 380.36 |  |  |
| 19 | 169.76 | 1161.16 (1161.71) | 10900.29 | 380.36 | 839.73 (839.3) | 7756.11 |
| 20 | 169.76 | 731.74 | 7845.53 | 380.36 | **829.98** (829.05) | 7268.06 |
| 21 | “ | 722.64 | 7721.48 | ” | 840.38 | 7316.56 |
| 22 | “ | 721.39 | 7724.56 | ” | 853.72 | 7303.56 |

note: it would seem that the PPL of the 1st part is nearly identical to the joint optimization experiment.

Does that mean that the FastText embeddings are not modifiable? It would be better if they were.

They are already modifiable, INFO : ('module.E', torch.Size([24689, 300]), torch.float32, True).

Let us see the epochs after the 2nd…

Best Valid-PPL on globals=380.22

# 2a: Loading pre-trained WT-2 model

To try to achieve better performance than 379 Valid-PPL on the globals’ task, we load the model already trained on WikiText-2 (with 184 Valid-PPL).

**def** load\_model\_from\_file():  
 saved\_model\_path = os.path.join(F.FOLDER\_NN, F.SAVED\_MODEL\_NAME)  
 model = torch.load(saved\_model\_path).module  
 logging.info(**"Loading the model found at: "** + str(saved\_model\_path))  
 **return** model

## Pretrained Simple GRUs on SemCor: necessary modifications

We can immediately spot a problem:

a model trained on a dataset with a different vocabulary has a different probability distribution, that is given by the softmax over the vocabulary.

INFO : Model:

INFO : DataParallel(

(module): RNN(

(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=**34377**, bias=True)

(linear2senses): Linear(in\_features=512, out\_features=**47936**, bias=True)))

Potential **problem n.2: the matrix containing the FastText embeddings** is based on WikiText-2’s vocabulary, not SemCor.

INFO : Parameters:

INFO : ('module.E', torch.Size([34377, 300]), torch.float32, True)

…

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

…

('module.linear2senses.bias', torch.Size([47936]), torch.float32, True)

However, I am not keeping a weights matrix at the start, I am selecting the globals in the current batch and index\_selecting their embeddings, that I pass on to the 1st layer of the GRU.

Therefore, I should be able to replace matrix E(WT-2) with E(SemCor) without issues.

Replacing it is necessary because the vocabulary of the 2 text corpuses is not identical.

Back on **problem n.1**:

('module.linear2global.weight', torch.Size([34377, 512]), torch.float32, True)

('module.linear2global.bias', torch.Size([34377]), torch.float32, True)

('module.linear2senses.weight', torch.Size([47936, 512]), torch.float32, True)

('module.linear2senses.bias', torch.Size([47936]), torch.float32, True)

I should replace the linear2global FF-NN that goes 512>34K with another: 512>25K. Same for senses (41K instead of 48K).

## Tests: using pre-trained model

### WT-2 to WT-2:

Loading a model that was trained on WikiText-2, and applying on WikiText-2 again. We should start from the good scores where we ended.

**The preliminary mini-experiment:**

Reading in the input text is again correct:

Batch globals: ['Markgraf', 'was', 'present', 'during', 'the', 'fleet', 'operation', 'that', 'resulted', 'in', 'the', 'Battle', 'of', 'Jutland', 'which', 'took', 'place', 'on', '31', 'May']

…

|  |  |  |  |
| --- | --- | --- | --- |
| *Epoch* | *Training PPL* | | |
|  | Globals | Senses | Multi-senses |
| 1 | 2450560215271.08 | - | - |
| 2 | 6239035980.76 |  |  |
| 3 | 1112.28 |  |  |
| 4 | 72.86 |  |  |
| 5 | 33.55 |  |  |
| 10 | 5.76 |  |  |
| 20 | 1.89 |  |  |
| 50 | 1.12 |  |  |

It overfits very quickly, needing only 20 epochs to become ~1 instead of about 150.

This supports the point that the pre-trained model has acquired knowledge about standard Language Modeling.

However, is there a way to avoid the abnormally high values for loss/PPL at the beginning? Is it due to gradient explosion?

Options:

* zero out any remaining gradients when loading the model
* use gradient\_clip for the first e.g.10 epochs.

Zeroing out any remaining gradients:

this may be already done…

\_grad\_fn is None and grad is None

And at the start of every batch, **for** b\_idx **in** range(len(train\_dataloader)-1): we already do:

optimizer.zero\_grad()

An example of gradient clipping:

optimizer.zero\_grad()

loss, hidden = model(data, hidden, targets)

loss.backward()

torch.nn.utils.clip\_grad\_norm\_(model.parameters(), args.clip)

optimizer.step()

I can also log the values of the gradient norm for the parameters, in the mini-experiment, both with and without using the pretrained model.

In the range: 1,17,31,104,253.

**Mini-experiment: from scratch, gradient value clipping <=100 for epochs 1-5**

|  |  |  |
| --- | --- | --- |
| *Epoch* | *Training PPL* | |
|  | Globals | (previous) |
| 1 | 32134.02 | 32134.02 |
| 10 | 107.07 | 107.07 |
| 100 | 94.63 | 94.63 |
| 150 | 1.71 | 1.71 |
| 200 |  | 1.08 |

identical. Gradient clipping has no effect here

**Mini-experiment: from pre-trained, gradient value clipping <=100 for epochs 1-5**

|  |  |  |
| --- | --- | --- |
| *Epoch* | *Training PPL* | |
|  | Globals | (previous) |
| 1 | 2450560215271.08 | ” |
| 2 | 6239035980.76 | ” |
| 3 | 1112.28 | ” |
| 4 | 72.86 | ” |
| 5 | 33.55 | ” |

identical.

Mini-experiment: from pre-trained, gradient value clipping <=10 for epochs 1-5:

identical

Mini-experiment: from pre-trained, gradient value clipping <=1 for epochs 1-5:

identical.

**Mini-experiment: from pre-trained, gradient norm clipping <=1 for epochs 1-5**

|  |  |  |
| --- | --- | --- |
| *Epoch* | *Training PPL* | |
|  | Globals |
| 1 | 2045326337365.45 |
| 2 | 1178056947.08 |
| 3 | 222.74 |
| 4 | 58.55 |
| 5 | 26.71 |

**Mini-experiment: from pre-trained, gradient norm clipping <=0.001 for epochs 1-5**

still out of order.

We do not know how we obtain this, however we know that gradient clipping is not useful to contain it.

Question: does it matter?

I can proceed with mini-experiment and experiment on SemCor, and see what happens there.

WT-2 to SemCor:

**The preliminary mini-experiment:**

|  |  |  |  |
| --- | --- | --- | --- |
| *Epoch* | *Training PPL* | | |
|  | Globals | Senses | Multi-senses |
| 1 | 23351.47 (24045.83) | 39967.08 (39522.51) | 40132.43 (40973.28) |
| 10 | 48.74 (51.83) | 91.16 (64.16) | 225.99 (158.73) |
| 25 | 3.23 | 36.27 | 55.03 |
| 100 | 1.06 (31.96) | 34.86 (34.89) | 52.88 (52.95) |
| 200 | 1.03 (3.26) | 34.84 (1.2) | 52.76 (1.25) |
| 270 | 1.03 | 1.65 | 1.7 |

The pre-trained model (2GRUs) holds knowledge about the standard LM task, and overfits sooner on it.

**The experiment:**

(GPUs 2 and 3)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Epoch* | *Training PPL* | | | *Validation PPL* | | |
|  | Globals | Senses | Multi-senses | Globals | Senses | Multi-senses |
| 1 | 1100.86 | 1156.81 | 10820.14 | 656.51 | 844.11 | 7793.7 |
| 2 | 490.58 | 732.01 | 7830.48 | 536.48 | **830.65** | 7250.19 |
| 3 | 396.29 | 722.84 | 7720.21 | 472.76 | 840.68 | 7285.01 |
| … |  |  |  |  |  |  |
| 13 | 188.06 | 650.04 | 6355.3 | 370.42 | 908.32 | 'correct\_all\_s': 6075, 'top\_k\_all\_s': 28283 |
| 14 | 179.85 | 640.28 | 6195.91 | 369.73 | 918.21 | 7703.11 |
| 15 | 172.28 | 633.14 | 6070.25 | **369.56** | 922.97 | 7816.57 |
| 16 | 165.25 | 638.45 | 6191.06 | 931.92 | 931.92 | 7859.16  'correct\_multi\_s': 0, 'top\_k\_multi\_s': 1471, |

Current results:

Globals-ValidPPL = **369.56** @ epoch 15,

Senses-ValidPPL = 830.65 @ epoch 2,

best senses stats = 6077 & 28283 / 80640, 0 & 1471 / 25345

Comparison with the previous experiment (Simple GRUs on SemCor alone). We had:

Globals-ValidPPL=**379.84** @epoch 18,

Senses-ValidPPL=829.05 @ epoch 2,

best senses stats: 6077 & 28536 / 80640, 0 & 1471 / 25345.

In the interest of research:

what is the overlap between WT-2’s vocabulary and SemCor’s vocabulary?

# 3: Sense context average

### Plan

Must: compute the average of the preceding context of each sense (hyperparameter: -5 ,-10, --15, -20).

It’s not a proper Skip-gram, because that mechanism would entail trying to predict the surrounding context with a NN architecture to obtain representations for the senses.

The purpose is to compare the context average around a sense s\_i with the representation (that can again be an average) of the context C.

Careful: the cosine distance between average of the sense context and average of the current context was used by J.Resinger & R.J.Mooney 2010 - "Multi-Prototype Vector-Space Models of Word Meaning". It must be cited.

Although what they did was:

- Represent the contexts  
- Cluster them  
- The resulting cluster centroids will be the word sense embeddings

It will be useful to review the multi-sense literature to understand which methods are similar to ours.

Plan of both the context-average method and the self-attention scores method:

* We keep a running average of the context, i.e. the last C word embeddings we encounter in the global task.
* For every sample s in batch\_size \* sequence\_length,   
   Individuate the K most likely predicted globals, select their senses  
   Compute the cosine similarity between the selected senses and the running average  
   of the word embeddings up to s  
   Rank the selected senses based on the cosine similarity
* observation: the specific meaning of a sense in a context is not necessarily close to the average of the context, in embedding terms
* instead of the sense embedding, I can use the context in which sense embeddings occur, like Skip-Gram (which means that I have to read the training set beforehand and register the past context of each sense)

### Implementation

We have the senses only in the SemCor corpus. In version 1.0, we will use that.

Although it is true that we could use WikiText-2 to help the dummySenses.

Read the SemCor corpus.

Every time we encounter a sense *s*, we add the average of the last i=5,10,15,20 word embeddings (i.e. FastText) to a storage.

We also update the counter of how many times we encountered a sense.

At the end of the corpus, divide (sum of sense-context averages) / (number of occurrences of that sense)

Sometimes, when we try to divide, num\_occurrences is 0. Why?

Because we are operating with a mini-experiment on a fragment of SemCor, and it doesn’t have the whole vocabulary. Adding corner case.

### Tests

We run a mini-experiment and see whether the computation of the sense-context-averages yields correct results or not.

sense\_idx=13859 ;

context\_stacked[:,0:3]=[[ 0. 0. 0. ]

[-0.05174419 0.07396396 -0.01305688]

[ 0.01587803 -0.0225705 -0.07099194]]

current\_context\_avg= [-0.01793308 0.02569673 -0.04202441]

# note: the average is weighted to exclude “0” vectors,

so we have 0.073 – 0.022 =~0.5, /2=0.25, which is correct

Occurrence number 2:

sense\_idx=13859 ;

context\_stacked[:,0:3]= [[-0.03145653 0.0327874 0.03962896]

[-0.05174419 0.07396396 -0.01305688]

[ 0.01587803 -0.0225705 -0.07099194]]

current\_context\_avg=[-0.0224409 0.02806029 -0.01480662]

and then

INFO : sense\_idx=i =13859

sense\_ctx\_avgs\_A[i,0:3] =[-0.02018699 0.02687851 -0.02841551]

passed.