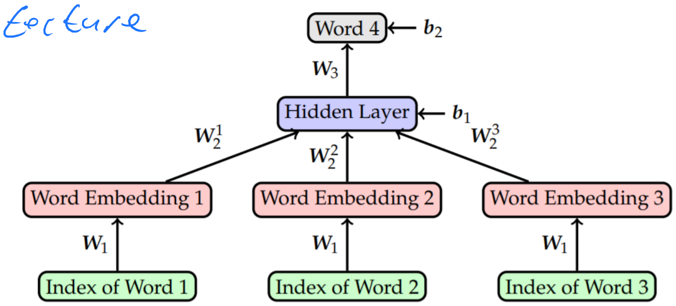
1. Network Architecture



The network consists of an input layer, embedding layer, hidden layer and output layer. The input consists of a sequence of 3 consecutive words, provided as integer valued indices, i.e., the 250 words in our dictionary are arbitrarily assigned a unique integer between 0 and 249. The embedding layer maps each word to its corresponding vector representation. This layer has $3 \times d$ units, where d is the embedding dimension, and functions as a look-up table. We will share the same look-up table for all the 3 positions, so we will learn a single common word embedding matrix for each context position. The embedding layer is connected to the hidden layer, which uses a sigmoid loss activation function. The hidden layer is connected to the output layer, and the output layer is a softmax over the 250 words in our dictionary.

1. The trainable parameters of the model consist of 3 weight matrices and two bias vectors. Assuming that we have 250 words in the dictionary, use three words as our input context, a 16-dimensional word embedding and a hidden layer with 128 units. What is the total number of trainable parameters in the model? Which part of the model has the highest number of parameters?

Looking at the code, we use one-hot encoding to map the words to vectors. So

$W_1 \in \mathbb{R}^{d \times m}$ and no bias vector. (m = # words in vocabulary)

So we have $\boxed{md \text{ learnable parameters}}$ in the embedding. It is simply a lookup table where the i-th column of W_1 is the vector corresponding to the i-th word in the dictionary.

The hidden layer looks like this:

$$y_1 = W_2 \cdot x + b_1$$

with $x \in \mathbb{R}^{3d}$ and $W_2 \in \mathbb{R}^{n \times 3d}$, $b \in \mathbb{R}^n$ where n is the number of hidden units.

So we have $\boxed{(3d+1)n \text{ trainable parameters}}$ in the hidden layer.

The sigmoid activation $h = sig(y_1)$ has no trainable parameters.

We then have W_3 to transform to the vocabulary size for the softmax operation. The softmax operation itself has no trainable parameters.

So: $p = \text{softmax}(W_3 h + b_2)$

with $h \in \mathbb{R}^n$, $W_3 \in \mathbb{R}^{m \times n}$, $b_2 \in \mathbb{R}^m$

where m is the vocabulary size.

So $\boxed{m(n+1) \text{ trainable parameters.}}$

And the output $p \in \mathbb{R}^m$, where p_i is the probability for the next word to be the one corresponding to index i.

1. The trainable parameters of the model consist of 3 weight matrices and two bias vectors. Assuming that we have 250 words in the dictionary, use three words as our input context, a 16-dimensional word embedding and a hidden layer with 128 units. What is the total number of trainable parameters in the model? Which part of the model has the highest number of parameters?

With those numbers: $m = 250$, $d = 16$, $n = 128$

So in total we have:

$$
\begin{aligned}
& md + (3d+1)n + (n+1)m \\
= \; & 250 \cdot 16 + (3 \cdot 16 + 1) \cdot 128 + (128+1) \cdot 250 \\
= \; & \underbrace{4000}_{\text{embedding}} + \underbrace{6272}_{\text{hidden}} + \underbrace{32250}_{\text{softmax}} = \underbrace{42522}_{\text{total}}
\end{aligned}
$$

So 42522 total parameters, most of which are in the softmax part with 32250 parameters.

3. Analysis

Using these methods, answer the following questions, each of which is worth 1 point.

1. Pick three words from the vocabulary that go well together (for instance, `'government of united'`, `'city of new'`, `'life in the'`, `'he is the'` etc.). Use the model to predict the next word. Does the model give sensible predictions? Try to find an example where it makes a plausible prediction even though the 4-gram wasn't present in the dataset (`raw_sentences.txt`).

2. Plot the 2-D visualization using the method `Model.tsne_plot.py`. Look at the plot and find a few clusters of related words. What do the words in each cluster have in common?

3. Are the words `'new'` and `'york'` close together in the learned representation? Why or why not?

4. Which pair of words is closer together in the learned representation: (`'government'`, `'political'`) or (`'government'`, `'university'`)? Why do you think that is?

1. Here are a few cases:

```
government of united states Prob: 0.52452
government of united people Prob: 0.13137
government of united ? Prob: 0.03084
government of united days Prob: 0.01394
government of united school Prob: 0.01326
government of united . Prob: 0.01181
government of united like Prob: 0.01150
government of united , Prob: 0.00975
government of united life Prob: 0.00942
government of united work Prob: 0.00917
```

```
city of new york Prob: 0.79750
city of new . Prob: 0.02406
city of new ? Prob: 0.02314
city of new life Prob: 0.01275
city of new world Prob: 0.01049
city of new home Prob: 0.00810
city of new people Prob: 0.00804
city of new , Prob: 0.00773
city of new children Prob: 0.00698
city of new family Prob: 0.00642
```

```
life in the world Prob: 0.31910
life in the united Prob: 0.04173
life in the office Prob: 0.03706
life in the house Prob: 0.03606
life in the city Prob: 0.03477
life in the end Prob: 0.03303
life in the school Prob: 0.03227
life in the time Prob: 0.03169
life in the way Prob: 0.03020
life in the police Prob: 0.02886
```

```
he is the best Prob: 0.25396
he is the only Prob: 0.12350
he is the right Prob: 0.08501
he is the first Prob: 0.05613
he is the president Prob: 0.05043
he is the last Prob: 0.04568
he is the same Prob: 0.04010
he is the man Prob: 0.02826
he is the children Prob: 0.02783
he is the one Prob: 0.02570
```

```
this is the best Prob: 0.37305
this is the only Prob: 0.07927
this is the way Prob: 0.07203
this is the one Prob: 0.05942
this is the last Prob: 0.05412
this is the new Prob: 0.02865
this is the world Prob: 0.01974
this is the business Prob: 0.01887
this is the first Prob: 0.01856
this is the right Prob: 0.01636
```

```
we are the best Prob: 0.38957
we are the only Prob: 0.12695
we are the one Prob: 0.05357
we are the man Prob: 0.04066
we are the same Prob: 0.03722
we are the first Prob: 0.02882
we are the police Prob: 0.02845
we are the last Prob: 0.01913
we are the at Prob: 0.01728
we are the president Prob: 0.01673
```

```
people of the world Prob: 0.19442
people of the police Prob: 0.12672
people of the united Prob: 0.10829
people of the day Prob: 0.05903
people of the best Prob: 0.04363
people of the one Prob: 0.04277
people of the team Prob: 0.03642
people of the house Prob: 0.02075
people of the time Prob: 0.02047
people of the work Prob: 0.01977
```

```
the people are going Prob: 0.29415
the people are good Prob: 0.14418
the people are . Prob: 0.07307
the people are in Prob: 0.02917
the people are not Prob: 0.02745
the people are out Prob: 0.02728
the people are the Prob: 0.02708
the people are still Prob: 0.02073
the people are right Prob: 0.01716
the people are about Prob: 0.01650
```

```
today , i said Prob: 0.13656
today , i know Prob: 0.10425
today , i do Prob: 0.09944
today , i want Prob: 0.07825
today , i would Prob: 0.05340
today , i have Prob: 0.04638
today , i think Prob: 0.04321
today , i just Prob: 0.04011
today , i say Prob: 0.03668
today , i should Prob: 0.03115
```

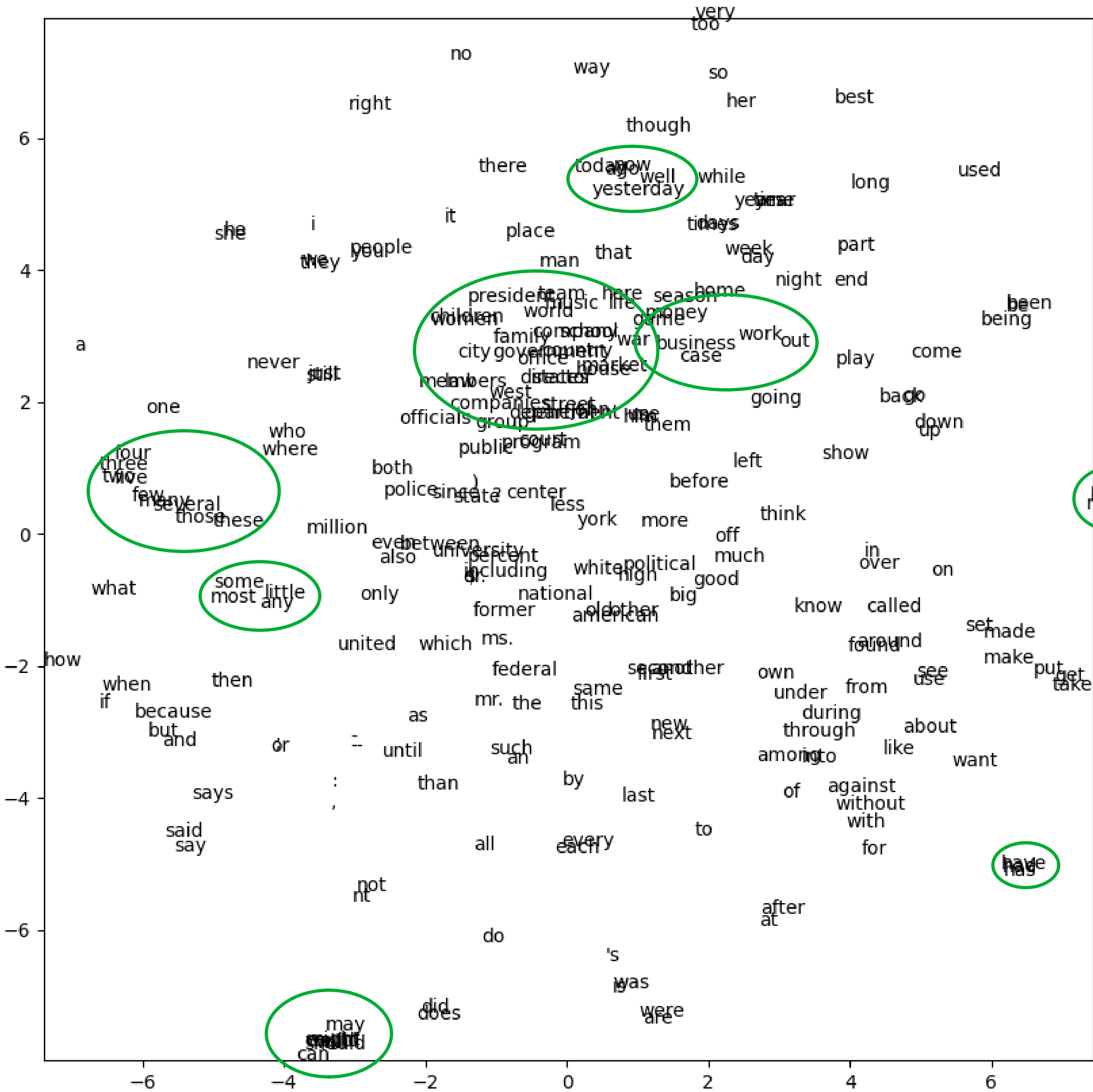
```
today i want to Prob: 0.58510
today i want it Prob: 0.09032
today i want you Prob: 0.04210
today i want them Prob: 0.03963
today i want me Prob: 0.03310
today i want . Prob: 0.02135
today i want more Prob: 0.01757
today i want him Prob: 0.01208
today i want , Prob: 0.01128
today i want this Prob: 0.01012
```

Actually, most of those are not in the dataset!

both not in dataset

In general, the model makes very sensible predictions!

2.



Obviously, there are many more!

There are many interesting clusters here, I looked at a few using the predict_next_word method:

```
Words closest to 'their':
my: 2.3766517639160156
your: 2.6984517574310303
our: 2.7101080417633057
his: 2.958097457885742
its: 3.1767351627349854
mr.: 5.300422191619873
own: 5.328340530395508
american: 5.54339599609375
national: 5.63131046295166
political: 5.667181491851807
```

```
Words closest to 'government':
family: 1.398392915725708
states: 1.5834486484527588
school: 1.813835859298706
west: 1.87473464012146
country: 1.8953932523727417
company: 1.9117995500564575
director: 2.04679536819458
general: 2.131866216659546
him: 2.2346177101135254
city: 2.2542688846588135
```

```
Words closest to 'business':
market: 2.1543092727661133
season: 2.155348062515259
states: 2.2388744354248047
street: 2.272861957550049
war: 2.2784671783447266
money: 2.3131847381591797
game: 2.3132553100585938
country: 2.3956894874572754
us: 2.456209897994995
world: 2.467219591140747
```

```
Words closest to 'could':
should: 1.5689303874969482
would: 1.964989185333252
might: 2.6491379737854004
will: 3.6709797382354736
may: 3.695051431655884
can: 4.829795837402344
nt: 5.843173027038574
since: 5.94174861907959
we: 6.113143444061279
did: 6.200811862945557
```

```
Words closest to 'not':
nt: 2.6908345222473145
also: 3.261019229888916
even: 4.108604907989502
between: 4.506925106048584
officials: 4.781681060791016
both: 4.872125625610352
police: 4.910123825073242
percent: 5.097867012023926
?: 5.112765312194824
): 5.129517078399658
```

```
Words closest to 'have':
had: 2.803586959838867
has: 3.2401018142700195
under: 5.239180564880371
into: 6.172253131866455
place: 6.263750076293945
through: 6.300256252288818
among: 6.349918842315674
less: 6.4732136726379395
between: 6.624838352203369
without: 6.653467655181885
```

```
Words closest to 'put':
set: 4.316836833953857
made: 4.7188496589660645
make: 5.001589298248291
get: 5.293972969055176
see: 5.869054794311523
around: 5.882547855377197
take: 5.990849018096924
called: 6.020701885223389
use: 6.197763442993164
about: 6.274527072906494
```

```
Words closest to 'without':
against: 3.0776023864746094
with: 3.515408515930176
about: 4.079280376434326
for: 4.1126885414123535
during: 4.179197311401367
of: 4.223660469055176
found: 4.28851842880249
after: 4.405359268188477
among: 4.500378608703613
under: 4.635783672332764
```

```
Words closest to 'say':
said: 4.105088233947754
says: 4.133743762969971
american: 6.347987174987793
me: 6.387940406799316
yesterday: 6.407471656799316
house: 6.423730850219727
days: 6.5162506103515625
same: 6.52865743637085
before: 6.532398700714111
times: 6.54292631149292
```

```
Words closest to 'now':
ago: 2.6296496391296387
yesterday: 2.8062241077423096
today: 2.963059663772583
season: 3.4153084754943848
times: 3.495828151702881
street: 3.5382697582244873
well: 3.59248948097229
us: 3.67444109916687
states: 3.675248384475708
then: 3.706301212310791
```

```
Words closest to 'four':
three: 1.9659196138381958
five: 2.4033846855163574
several: 3.152776002883911
two: 3.6519858837127686
few: 3.859572172164917
many: 3.9678232669830322
those: 4.631694793701172
million: 4.644901275634766
one: 4.778314590454102
university: 5.6221089363098145
```

```
Words closest to 'some':
most: 2.721904993057251
several: 4.0428338050842285
those: 4.097591400146484
many: 4.214697360992432
million: 4.274576663970947
little: 4.299197673797607
university: 4.549999713897705
former: 4.595312118530273
state: 4.6576619148254395
few: 4.85149621963501
```

What those words have in common is that they often appear before/after/between $\boxed{\text{(same context)}}$ the same other words (not each other).

So we see, for example, pronouns clustering together. Also nouns, and within these those with a similar meaning. Or words that describe time, and so on.

3.

```
distance 'new' and 'york': 7.920381546020508
```

They are not! Simply because they often appear after each other, not in place of each other. The order of words is learned in the other layers, not in the embedding.

4.

```
distance 'government' and 'political': 3.4061052799224854
distance 'government' and 'university': 3.9237935543060303
distance 'political' and 'university': 4.83165979385376
```

closest!

All 3 are related when it comes to meaning.

I think $||\text{government} - \text{political}|| < ||\text{government} - \text{university}||$ because "government" is more commonly found in the same context with "political" than with "university" in the given dataset.