CS 533: Natural Language Processing **(Due: 02/18/20)**

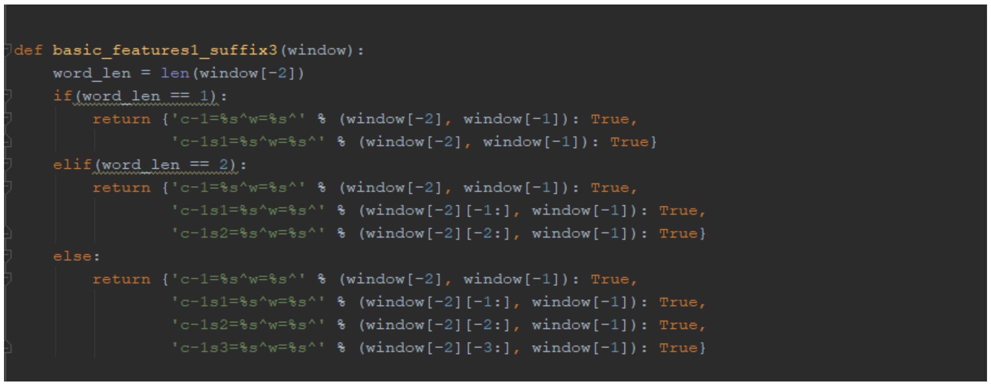
Assignment 2

Name: Prakruti Joshi, *NetID:* phj15 *Students discussed with:*

Problem 1: Log-linear Models

((2 + 2 + 2 + 2 + 4 + 2 + 2 + 2 (+10) = 18 (+10) points))

1. Implementation of $basic\_feature1\_suffix3$:



```python
def basic_features1_suffix3(window):
    word_len = len(window[-2])
    if(word_len == 1):
        return {'c-1=%s^w=%s^' % (window[-2], window[-1]): True,
                'c-1s1=%s^w=%s^' % (window[-2], window[-1]): True}
    elif(word_len == 2):
        return {'c-1=%s^w=%s^' % (window[-2], window[-1]): True,
                'c-1s1=%s^w=%s^' % (window[-2][-1:], window[-1]): True,
                'c-1s2=%s^w=%s^' % (window[-2][-2:], window[-1]): True}
    else:
        return {'c-1=%s^w=%s^' % (window[-2], window[-1]): True,
                'c-1s1=%s^w=%s^' % (window[-2][-1:], window[-1]): True,
                'c-1s2=%s^w=%s^' % (window[-2][-2:], window[-1]): True,
                'c-1s3=%s^w=%s^' % (window[-2][-3:], window[-1]): True}
```

Figure 1: Implementation of $basic\_feature1\_suffix3$ that extracts features in $basic\_features1$ plus suffixes of length up to 3 of window[-2].

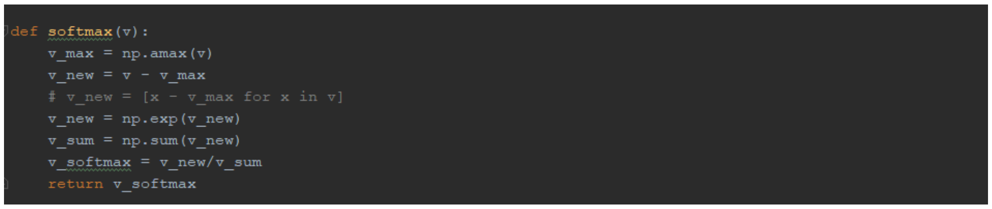
2. Number of feature types extracted:

   (a) $basic\_features1 = 61138$
   (b) $basic\_features2 = 233869$
   (c) $basic\_feature1\_suffix3 = 171678$

3. Figure 2 show the implementation of softmax transformation of vector. The modification of subtracting the maximum value of the vector from each value in the vector helps in avoiding overflow or underflow incase a particular value is too large or too small. The softmax of the modified vector remains the same as that of the original vector due to the following exponential property:

$$
\begin{aligned}
e^{(a-b)} &= e^a/e^b \\
&Thus, \\
e^{(a-z)}/(e^{(a-z)} + e^{(b-z)} + e^{(c-z)} + e^{(d-z)}) \\
&= e^a/e^z/(e^{(a-z)} + e^{b-z} + e^{c-z} + e^{d-z})/e^z \\
&= e^a/(e^a + e^b + e^c + e^d)
\end{aligned} \tag{1}
$$

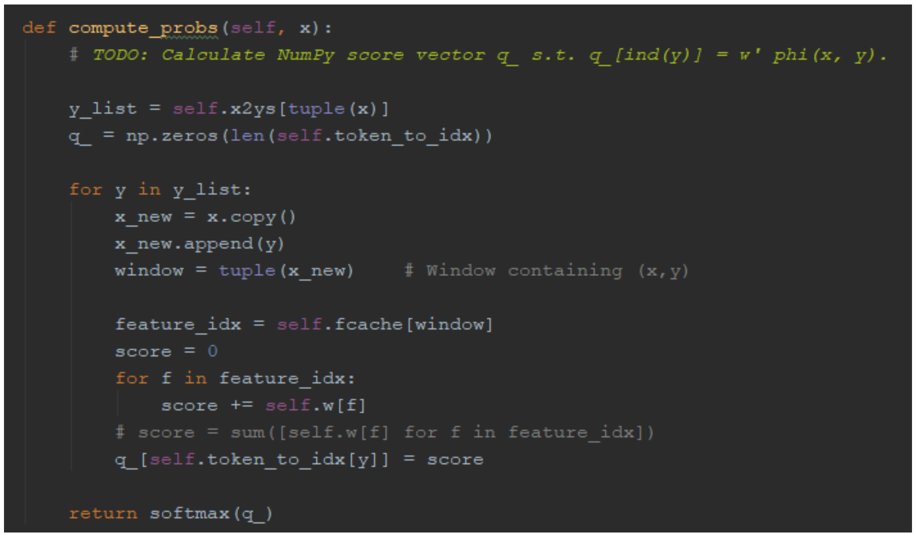
Due to the above property, subtracting a constant value from all the elements of vector will not affect the softmax output of the vector.



```
def softmax(v):
    v_max = np.amax(v)
    v_new = v - v_max
    # v_new = [x - v_max for x in v]
    v_new = np.exp(v_new)
    v_sum = np.sum(v_new)
    v_softmax = v_new/v_sum
    return v_softmax
```

Fig 2: Softmax transformation of the vector.

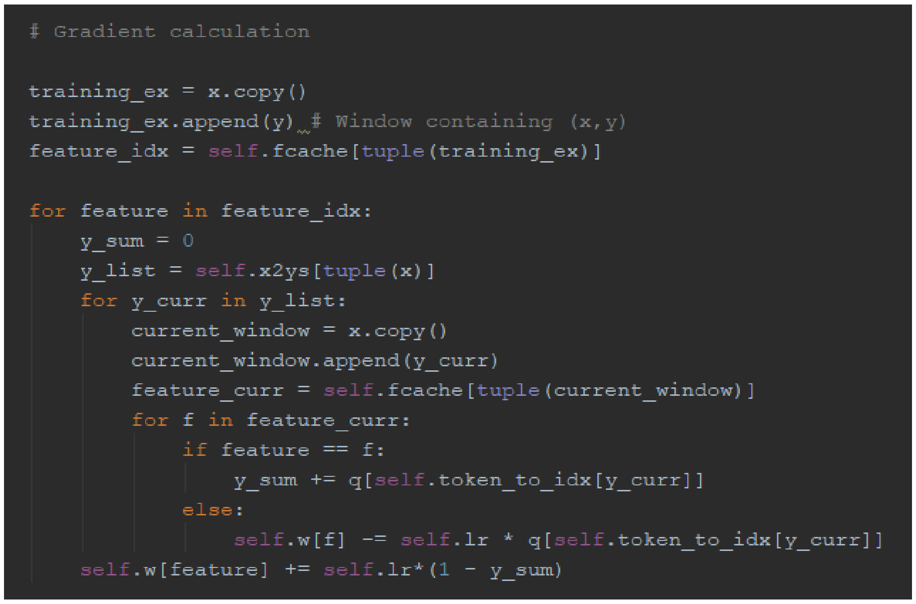
4. Implementation of compute_probs:



```
def compute_probs(self, x):
    # TODO: Calculate NumPy score vector q_ s.t. q_[ind(y)] = w' phi(x, y).

    y_list = self.x2ys[tuple(x)]
    q_ = np.zeros(len(self.token_to_idx))

    for y in y_list:
        x_new = x.copy()
        x_new.append(y)
        window = tuple(x_new)     # Window containing (x,y)

        feature_idx = self.fcache[window]
        score = 0
        for f in feature_idx:
            score += self.w[f]
        # score = sum([self.w[f] for f in feature_idx])
        q_[self.token_to_idx[y]] = score

    return softmax(q_)
```

Fig 3: Function compute_prob which generates the softmax values of the score vector given a context x.

5. Figure 4 shows the implementation of the gradient update.



```
# Gradient calculation

training_ex = x.copy()
training_ex.append(y) # Window containing (x,y)
feature_idx = self.fcache[tuple(training_ex)]

for feature in feature_idx:
    y_sum = 0
    y_list = self.x2ys[tuple(x)]
    for y_curr in y_list:
        current_window = x.copy()
        current_window.append(y_curr)
        feature_curr = self.fcache[tuple(current_window)]
        for f in feature_curr:
            if feature == f:
                y_sum += q[self.token_to_idx[y_curr]]
            else:
                self.w[f] -= self.lr * q[self.token_to_idx[y_curr]]
    self.w[feature] += self.lr*(1 - y_sum)
```

Fig 4: Gradient update implementation inside the method do_epochs

6. Figure 5 shows the validation perplexity for different learning rate. The optimal learning rate with *basic_features1* as the feature extractor comes out to be **1** with validation perplexity as **149.16** and training perplexity coming out to be **5.21**.

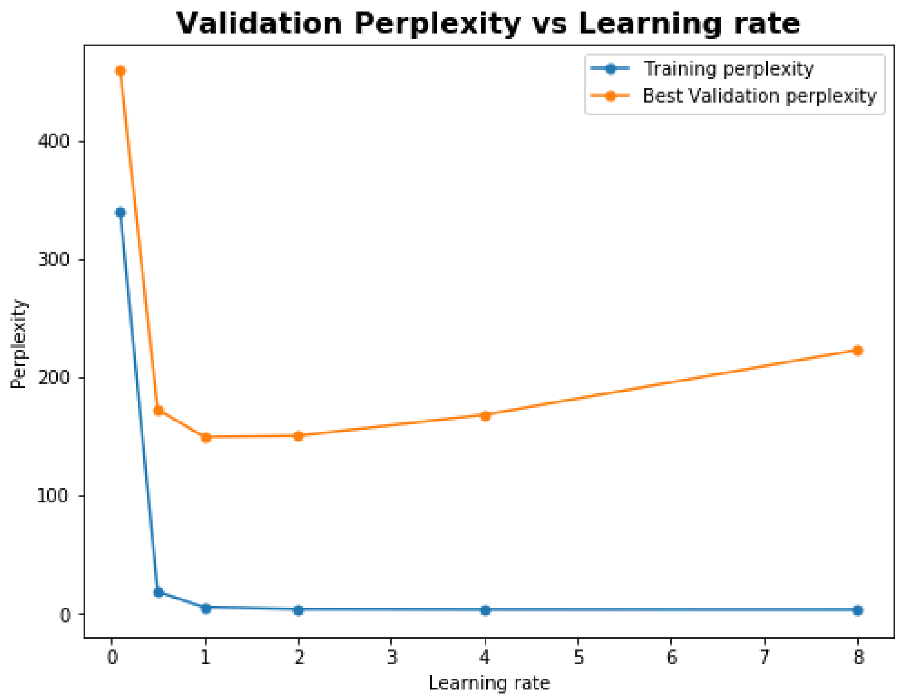
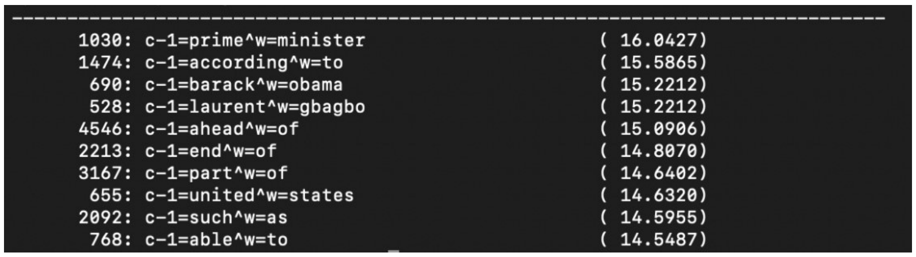


Fig 5: Validation and training perplexity for different learning rates

7. The validation perplexity using *basic_features1_suffix3*, learning rate as 0.25 and seed 42 is 79.14. This is lower than the model using *basic_feature1*. This is the best model so far. However, the features generated using this model are not useful for inference since it includes the suffix of the previous word.
   Thus, the best model using *basic_feature1* as the feature generator has learning rate as 1. The top 10 features produced by this model is shown in figure 6. These features have been assigned the highest weight. The weight given to a particular feature f represents the probability or the co-occurrence of the word given the context x which form this feature f. For example, the second feature in Figure 6 signifies the tuple (according, to). This implies that given the previous word as *according*, *to* has the maximum probability to occur. The higher the weight, the more is the probability of the feature occurring. Here in our case the feature is the current word and the previous word. Thus, we can observe that "obama" has higher probability of occurring if the context is "barack". This is intuitive as we often see "barack obama" together instead of just "barack". Similar example is that of united states. These feature weights are learned during training. The features which have a higher value out of the entire corpus are displayed in the Figure 6.



```
--------------------------------------------------------------------------------
      1030: c-1=prime^w=minister                             ( 16.0427)
      1474: c-1=according^w=to                               ( 15.5865)
       690: c-1=barack^w=obama                               ( 15.2212)
       528: c-1=laurent^w=gbagbo                             ( 15.2212)
      4546: c-1=ahead^w=of                                   ( 15.0906)
      2213: c-1=end^w=of                                     ( 14.8070)
      3167: c-1=part^w=of                                    ( 14.6402)
       655: c-1=united^w=states                              ( 14.6320)
      2092: c-1=such^w=as                                    ( 14.5955)
       768: c-1=able^w=to                                    ( 14.5487)
```

Fig 6: Top 10 features produced by using the best model

8. The validation perplexity for 10 random seed values (range: 1-10000) are as follows:

Seed	Validation perplexity
100	142.695126
1113	150.6685
5267	146.8731
375	142.2641
9	151.7314
9512	143.3741
216	151.6894
7804	151.3923
612	151.1102
8973	153.2028

Mean of validation perplexity: 148.5
Standard deviation of validation perplexity: 4.27

9. Bonus question (collaborated with Keya Desai- kd706, Twisha Naik- tn265 for this question) After trying different learning rates, seeds, feature extractors and training on 100 percent data, the best model has the following parameters. It is much better than above models and attains the following results:

   (a) Feature extractor: $basic\_feature1\_suffix3$

   (b) Learning rate: 0.25

   (c) Seed: 42

   (d) Number of epochs: 3

   (e) Validation perplexity: 99.1144

   Due to computational and time limitations, we have only run this model for 3 epochs. The results would be better if it was run for more epochs. Command to duplicate the results:
   python assignment2.py –lr 0.25 –train_fraction 1 –val_fraction 1 –features 'basic1suffix3'

Problem 2: Feedforward Neural Language Model

((2 + 2 + 2 + 2 + 2 + 2 = 12 points))

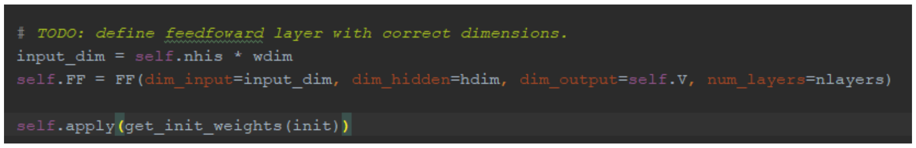
1. Figure 7 shows the correct definition of *FF* by specifying the correct dimensions of:

   (a) Input: It is the number of words conditioned on times the word embedding dimension (for each word) i.e. *self.nhis*wdim*

   (b) Hidden layer dimension: It is the *hdim* parameter.

   (c) Output: The dimension of the output is the vocab size i.e. *self.V* as the last layer produces the probabilities.

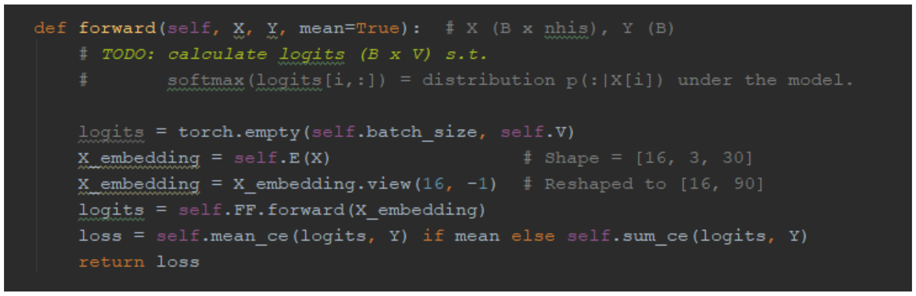
   (d) Number of layers in model: It is the *nlayers* parameter.



```
# TODO: define feedfoward layer with correct dimensions.
input_dim = self.nhis * wdim
self.FF = FF(dim_input=input_dim, dim_hidden=hdim, dim_output=self.V, num_layers=nlayers)

self.apply(get_init_weights(init))
```

Fig 7: The initialization of FF object

2. Figure 8 shows the implementation of forward method.



```
def forward(self, X, Y, mean=True):  # X (B x nhis), Y (B)
    # TODO: calculate logits (B x V) s.t.
    #       softmax(logits[i,:]) = distribution p(:|X[i]) under the model.

    logits = torch.empty(self.batch_size, self.V)
    X_embedding = self.E(X)                      # Shape = [16, 3, 30]
    X_embedding = X_embedding.view(16, -1)   # Reshaped to [16, 90]
    logits = self.FF.forward(X_embedding)
    loss = self.mean_ce(logits, Y) if mean else self.sum_ce(logits, Y)
    return loss
```

Fig 8: Definition of forward function

3. Training a linear layer using 3 previous words with batch size 16 up to 10 epochs. Keeping wdim=hdim and varying wdim as [1,5,10,100,200]. The **figure 9** shows the optimized perlexity for different learning rates [0.00001, 0.00003, 0.0001, 0.0003, 0.001]. For 0.001 and wdim = 100,200 ; the optimized perplexity is quite large rendering the figure 9 not that informative. Thus, for visualization purpose, I have clipped the maximum optimized perplexity to 1700. The **figure 10** represents the optimized perplexity after this modification. The optimal learning rate comes out to be **0.00003**.

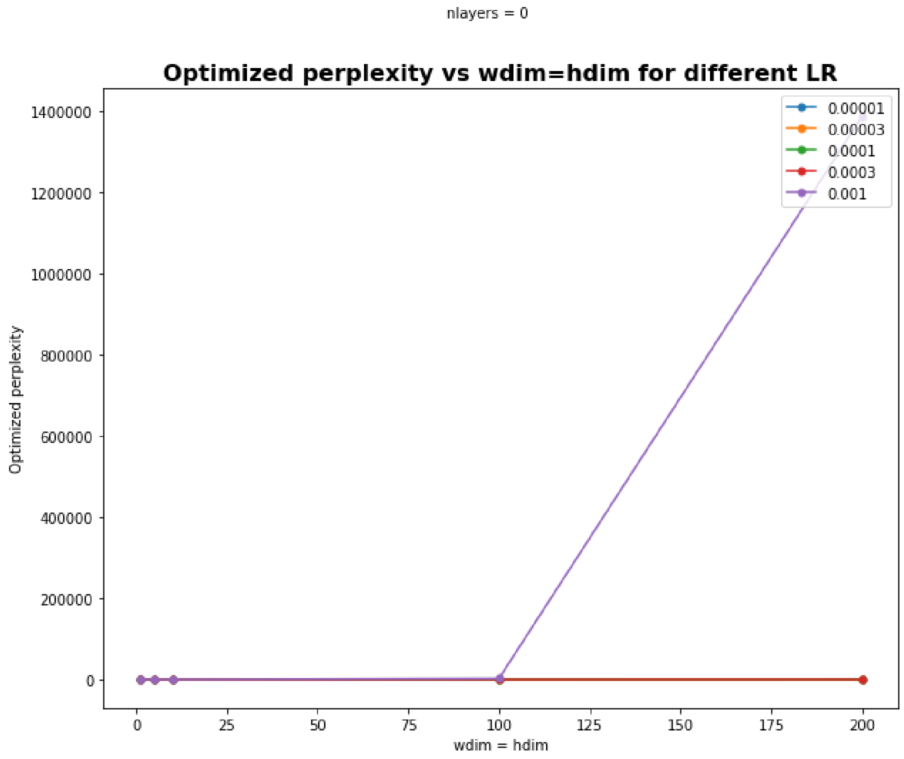


Fig 9: Optimized perplexity vs wdim [1,5,10,100,200] for different learning rates. Number of layers in the model us 0. No clipping is used.

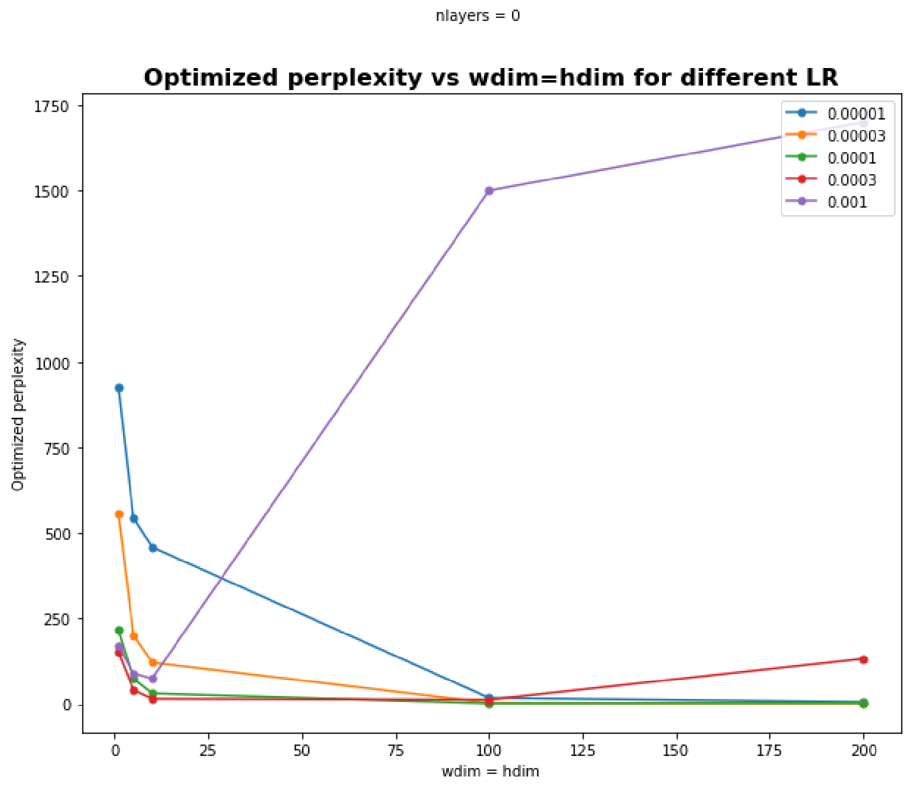


Fig 10: Optimized perplexity vs wdim [1,5,10,100,200] for different learning rates. Number of layers in the model us 0. Values larger than 1750 are clipped to 1750 for better visualition purposes. Value for learning rate = 0.001 and wdim = 200 had to be clipped.

4. The above analysis for number of layers = 1 can be seen in Figure 11.

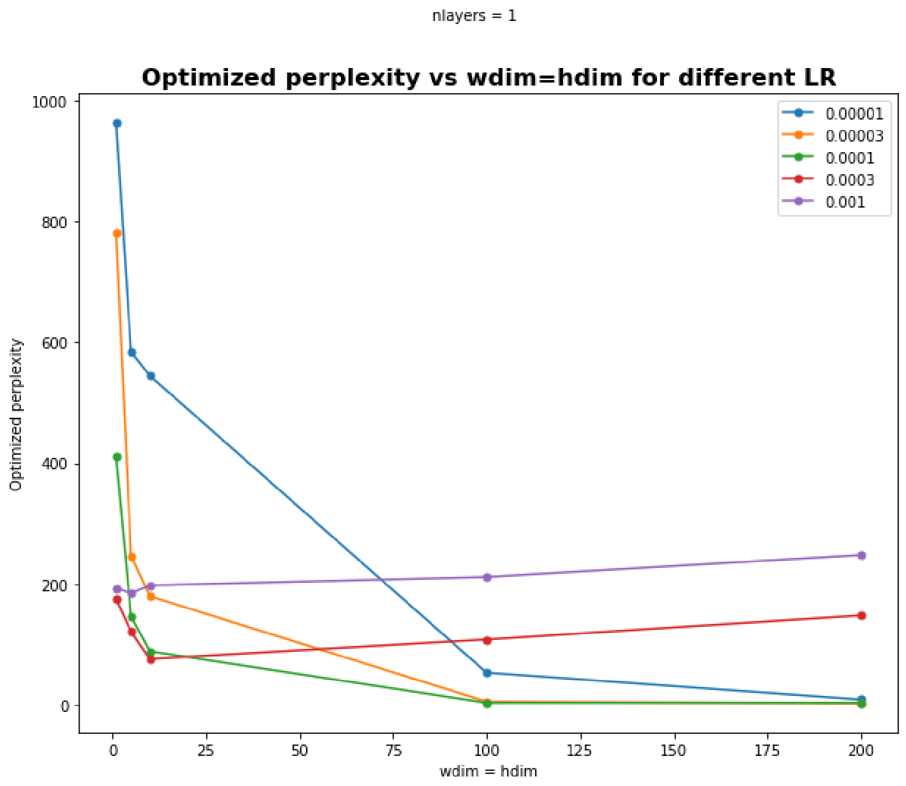


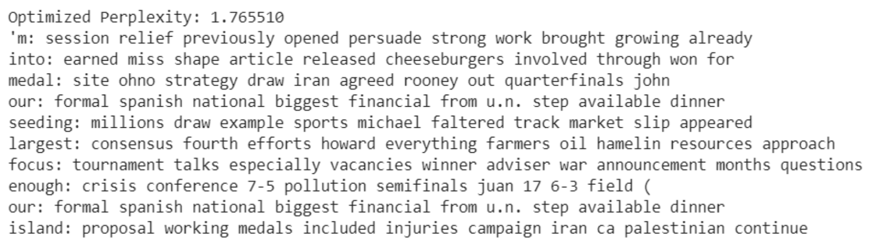
Fig 11: Optimized perplexity vs wdim [1,5,10,100,200] for different learning rates. Number of layers in the model is 1.

5. Running the linear and non-linear model using wdim = 30, number of epochs = 1000 and learning rate= 0.00003 :

nlayers	**Validation perplexity**	**Training perplexity**	**Epochs**	**Loss**
0	1.77	1.83	176	0.602
1	2.84	3.01	134	1.065

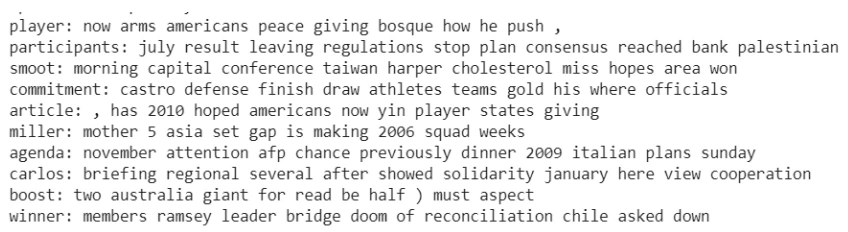
From the table, we can see that even after adding non-linearity, the model does not converge to a smaller loss and perplexity. The linear model takes slightly more epochs to converge but it converges to a smaller loss. This refutes the hypothesis that it takes more updates for bigger models to converge, but when they do they can achieve a smaller training loss.

6. Some of the examples of the nearest neighbors of trained word embeddings (in cosine similarity) are shown in Figure 12.
Using the cosine similarity, similar words should have similar word embeddings and thus, should appear as neighbors. Some of the examples of nearest neighbors of the trained words makes sense, however some seem irrelevant and do not intuitively make much sense. For example, in Fig 11-c, the nearest neighbors of the word *minister* are "administration, position, lead, welfare" which make sense in the context of minister in an administrative post. Also, in Fig 11-a,the word *medal* has "strategy, draw, quaterfinals" in the neighborhood which keeping sports as context make sense. However, in Fig 11-b, the nearest neighbors of the word *winner* are "doom, reconciliation" which do not make much sense. Thus, the pair of words do not always make much sense in terms of similarity, however considering a context and group of words, some examples do make sense. This might be due to our definition of the model. We are considering a window of three (tri-gram) and extracting features from that window. Thus, immediate previous context plays a major role in the word embedding. The word embeddings are learned accordingly during training. Thus, words occurring in same context frequently are deemed similar by the model.



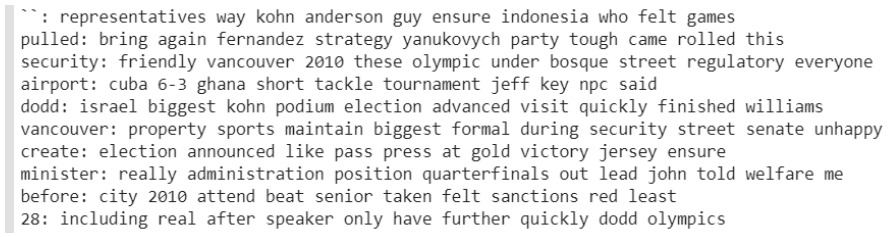
```
Optimized Perplexity: 1.765510
'm: session relief previously opened persuade strong work brought growing already
into: earned miss shape article released cheeseburgers involved through won for
medal: site ohno strategy draw iran agreed rooney out quarterfinals john
our: formal spanish national biggest financial from u.n. step available dinner
seeding: millions draw example sports michael faltered track market slip appeared
largest: consensus fourth efforts howard everything farmers oil hamelin resources approach
focus: tournament talks especially vacancies winner adviser war announcement months questions
enough: crisis conference 7-5 pollution semifinals juan 17 6-3 field (
our: formal spanish national biggest financial from u.n. step available dinner
island: proposal working medals included injuries campaign iran ca palestinian continue
```

Fig 11-a: Some interesting nearest neighbors of trained word embedding



```
player: now arms americans peace giving bosque how he push ,
participants: july result leaving regulations stop plan consensus reached bank palestinian
smoot: morning capital conference taiwan harper cholesterol miss hopes area won
commitment: castro defense finish draw athletes teams gold his where officials
article: , has 2010 hoped americans now yin player states giving
miller: mother 5 asia set gap is making 2006 squad weeks
agenda: november attention afp chance previously dinner 2009 italian plans sunday
carlos: briefing regional several after showed solidarity january here view cooperation
boost: two australia giant for read be half ) must aspect
winner: members ramsey leader bridge doom of reconciliation chile asked down
```

Fig 11-b: Some interesting nearest neighbors of trained word embedding



```
``: representatives way kohn anderson guy ensure indonesia who felt games
pulled: bring again fernandez strategy yanukovych party tough came rolled this
security: friendly vancouver 2010 these olympic under bosque street regulatory everyone
airport: cuba 6-3 ghana short tackle tournament jeff key npc said
dodd: israel biggest kohn podium election advanced visit quickly finished williams
vancouver: property sports maintain biggest formal during security street senate unhappy
create: election announced like pass press at gold victory jersey ensure
minister: really administration position quarterfinals out lead john told welfare me
before: city 2010 attend beat senior taken felt sanctions red least
28: including real after speaker only have further quickly dodd olympics
```

Fig 11-c: Some interesting nearest neighbors of trained word embedding